Alexander Stolpmann

An Intelligent Soft-Computing Texture Classification System

Pontypridd/Wolfenbüttel 2005

An Intelligent Soft-Computing Texture Classification System

Alexander Stolpmann

A submission presented in partial fulfilment of the
requirements of the University of Glamorgan / Prifysgol Morgannwg
for the degree of Doctor of Philosophy

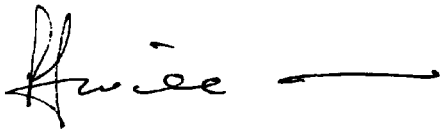
This research programme was carried out
in collaboration with the Fachhochschule Braunschweig/Wolfenbüttel

January 2005

Certificate of Research

This is to certify that, except where specific reference is made, the work described in this thesis is the result of the candidate. Neither this thesis, nor any part of it, has been presented, or is currently submitted, in candidature for any degree at any other university.

Candidate (Alexander Stolpmann)



Director of Studies (Prof. Robert J. Williams)

January 2005

Acknowledgements

The author likes to take the opportunity to mention and thank all those valuable people who by their assistance helped throughout this research work.

Thank you very much to my Director of Studies, Prof. Robert J. Williams. You made it possible with all your efforts in taking care of many formal necessities that this work reached the final stage. Prof. Williams followed my progress as second supervisor and took over to be the Director of Studies from Prof. Dr. Laurence S. Dooley, who left the University of Glamorgan to take a chair at Monash University in Melbourne, Australia. Thank you to Laurence Dooley for your valuable time and discussions. Thank you also to Prof. Dr. Phil A. Witting, head of the School of Electronics. Talking to you started the idea for a collaboration between the University of Glamorgan and the Fachhochschule Braunschweig/Wolfenbüttel. This gave not only me the chance for a very interesting research work.

On the other side of the Channel it was Prof. Dr. Klaus Harbusch, former vice-president of the Fachhochschule Braunschweig/Wolfenbüttel, who not only started the collaboration between the Universities, but also made it possible for me to follow my research whilst working for the Fachhochschule. Prof. Hans-Lothar Hanemann helped me to do my research and earn a living at the Fachbereich Informatik. This project would not have been possible without him. Thank you very much to both of you.

Prof. Dr. Jürgen Angele was my local supervisor before he left the Fachhochschule Braunschweig/Wolfenbüttel to start his own business. Thank you very much for our valuable discussions and for the chance to do real world projects.

Thank you to Prof. Dr. Friedhelm Seutter, Prof. Dr. Ulrich Klages, Prof. Dr. Wolfgang Schneider and all the members of the Fachbereich Informatik, who took an sincere interest in my work. Thank you to the colleagues at the library, the computer centre and the other

departments of the Fachhochschule. And last, but not least, thank you to Prof. Dr. Wolf-Rüdiger Umbach, the President of the Fachhochschule Braunschweig/Wolfenbüttel, for all your help.

Thank you to Dr. Frank Höwing, my *brother-in-arms* during this research. It always was and still is a pleasure to discuss matters, scientific or not, with you. And we had a great time in Wales.

Thank you to Gottfried Neumann-Olandt, probably the most important industrial partner I met. You showed me how image processing in industry works.

Finally a very big thank you to my parents, Christa and Hans-Jürgen Stolpmann. You always encouraged me to go my way and do my research, from the very early childhood days until today. And you started my anglophilia.

Abstract

The aim of this research work was to obtain a system that classifies texture. This so called Texture Classification System is not a system for one special task or group of tasks. It is a general approach that shows a way towards real artificial vision.

Finding ways to enable computerised systems to visually recognise its surroundings is of increasing importance for the industry and society at large. To reach this goal not only objects but less well describable texture has to be identified within an image.

To achieve this aim a number of objectives had to be met. At first a review of how natural vision works was done to better understand the complexity of visual systems. This is followed by a more detailed definition of what texture is. Next a review of image processing techniques, of statistical methods and of soft-computing methods was made to identify those that can be used or improved for the Texture Classification System.

A major objective was to create the structure of the Texture Classification System. The design presented in this work is the framework for a multitude of modules arranged in groups and layers with multiple feedback and optimisation possibilities.

The main achievement is a system for texture classification for which natural vision was used as a "blue-print". A more detailed definition of what texture is was made and a new texture library was started. The close review of image processing techniques provided a variety of applicable methods, as did the review and enhancement of statistical methods. Some of those methods were improved or used in a new way. Neural networks and fuzzy clustering were applied for classification, while genetic algorithms provide a means for self optimisation.

The concepts and methods have been used for a number of projects next to texture classification itself. This work presents applications for fault detection in glass container manufacturing, quality control of veneer, positioning control of steel blocks in a rotation oven, and measurement of hair gloss.

With the Texture Classification System a new, holistic approach for complex image processing and artificial vision tasks is being contributed. It uses a modular combination of statistics, image processing and soft-computing methods, easily adaptable to new tasks, includes new ideas for high order statistics, and incorporates self optimisation to achieve lean sub-systems. The system allows multiple feedbacks and includes a border detection routine. The new texture library provides images for future work of researchers.

Still a lot of work has to be done in the future to achieve an artificial vision system that is comparable to the human's visual capabilities. This is mainly due to the fact of missing computational resources. At least another decade of hardware development is needed to reach this goal. During this time more, better or even novel methods will be added to the Texture Classification System to improve its universal capabilities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background of the Work

Artificial vision – or how to make a computer see its environment. That is the main idea behind this research work.

This work started with the aim to bring together a number of research fields, namely the three mayor fields in soft-computing, i. e. fuzzy logic, neural networks and genetic algorithms, and industrial and scientific image processing. The important point is that the work has not brought forth a special tool for a special task, but an adaptive system.

1.2 Key Problems and Existing Solutions

The problems in artificial vision in general are the complexity of image processing task that have to be handled. Compared to the human visual system today's computers are no match by far. That means it is not possible to find solutions by using sheer computational power.

Today most industrial image processing systems are used for rather simple quality control or OCR tasks. These are usually "one-shot-solutions"[1] optimised for speed and used in a well defined environment. Components of such systems are very often binarisation followed by either edge detection or blob analysis and feature classification by object position or size.

[1]That are systems without an internal control mechanism through feedback of results for adjustment purposes.

For textures this approach will not work. Many researchers working on texture classification therefore use more sophisticated pre-processing, feature extraction and classification methods. All those approaches found in literature still use the "one-shot" design with no or very little information feedback.

Next to the purely academic approaches in classifying the BRODATZ textures, which are not optimally suited for real world problems (cf. App. A.1), an increasing number of researchers use texture classification for satellite image segmentation [BO02, CHL98, DG02, LXH98, MGF02, PL00, RZC+98]. Other applications include soil classification [Zha04], robotics [CANS03], classification of marble [dVA99], quality control of paper surface [MTP03] and image retrieval [PM94, dVA99]. A number of research works address special texture variations, i. e. 3D-texture [Wu03], coloured texture [IMKF03, LST04, LT04, MP04] and effects of illumination [Bar04, Gon02, Ran97].

Among those research works the majority uses the wavelet transform for texture classification [CANS03, CHL98, IMKF03, LST04, LT04, LXH98, Zha04] while the applications vary. Among the others there is a great variation of approaches and methods used, e. g. DellíAcqua and Gamba [DG02] use cooccurrence texture measures, Bashar and Ohnishi [BO02] a block processing approach inspired by human vision, Gonzalez [Gon02] and Randen [Ran97] apply filter functions for feature extraction, and Chen et al. [CHL98] and Martins et al. [MGF02] classify with neural networks.

All those systems work well for the special task they aim at. None of those systems aims at a more global solution for texture classification. Only Paget and Longstaff [PL00] attempt with their *open-ended texture classification* to include texture not known to the system beforehand. They use clustering of statistical data for classification.

So far no system is known to the author that holds the potential to embrace numerous approaches, i. e. all those mentioned above, for image preprocessing, feature extraction and classification all at the same time. Additionally no system is known that uses feedback within the system. All known systems use a feed-forward, non-parallel design. Again only the author's system design incorporates routines for automatic optimisation using genetic algorithms.

All the approaches mentioned above, valuable as they are, have the disadvantage of being feed-forward only. Additionally only very few if not single feature extraction and classification methods are being used. Those layouts will yield lean systems which are

well suited for specialised, well defined task. For a more general approach they are not expedient.

1.3 Aim and Objectives

The aim is to create an adaptable system for texture classification. At the same time this system and its design has to have the potential to be used for other related image processing tasks.

Utilising ideas created by nature over millions of years brings forth a system consisting of numerous specialised modules combined in a self-optimising structure. In such a system many new ideas can be included and used in parallel with other proven methods.

This leads to the idea to design a massively parallel and modular system that includes not only feedback possibilities but also self-optimisation routines. That again means that many different methods for pre-processing, feature extraction and classification, as well as optimisation methods have to tested and, if possible, improved for the use in the Texture Classification System described in this work.

All that said before promotes the question what such a system is good for.

- ▷ Looking at the work from a short distance it can be said that the aim is to design a system for texture classification. This might sound like a simple task, but it is not if real life texture images are being used instead of high quality studio photographies. The problems that arise will be discussed and solutions given in this work.

- ▷ Looking at the work from slightly away, the Texture Classification System shows its potentials to be used for all sorts of complex image processing and classification tasks. Due to its modular design it can be adapted for a variety of problems, especially in quality control. Examples are given in Chapter 4.

- ▷ Stepping back one step further it becomes clear, that this work is an important part on the way towards real artificial vision. There is still another decade, or two, before computer technology and computing methods will reach or even better the capabilities of the human visual system. Still it is important to start the development of such systems today.

To create the Texture Classification System a number of objectives have to be met.

- ▷ At first a close review of how natural vision works has to be made. This is needed to better understand the complexity of the human's visual system and to be able to derive the concepts for the creation of the Texture Classification System.
- ▷ A better and more refined definition of what texture is shall be given. Problems arising from existing texture image libraries, mainly image resolution, missing colour, unnatural contrast and limited choice of textures have to be dealt with.
- ▷ A review of image processing techniques has to take place. Existing methods have to be scrutinised in respect to the use in the Texture Classification System.
- ▷ Statistical methods of first and second order shall be applied. With first order statistics the emphasis has to be laid on higher moments. Second order statistics are added with a number of new approaches, i. e. wide-, multi-dimensional, and logarithmic cooccurrence matrices.
- ▷ A review of soft-computing methods in the light of creating classification modules for the Texture Classification System. Additionally the potential of soft-computing methods for optimisation tasks shall be tested.
- ▷ The design of a texture classification system. This system shall be constructed of interchangeable modules. Important are the feedback and optimisation possibilities within the system.
- ▷ Finally the design ideas and principles shall be tested with example applications.

As said above the Texture Classification System can be approached with different goals in mind. This can be the task of classifying texture, building a complex still versatile image processing system, or moving a step closer towards real artificial vision.

Keeping that in mind, a system has to be constructed that is capable to handle and make use of numerous image processing, feature extraction and classification techniques. If possible new techniques will be added by modifying or enhancing existing algorithms. It also has to be possible to add or exchange modules in the future to be able to include new or better algorithms.

An automatic optimisation unit has to be included into the system. This will make it possible to adjust parameter setting or choice of modules according to the quality of the system's output.

The system and its design ideas shall be tested with real world problems, as far as computer hardware technology allows.

Additionally problem arising from existing texture collections shall be solved.

To cover the subjects addressed in this work a great variety of research fields have to be taken into account. It starts at traditional image processing, transformation techniques, statistical evaluation, to all areas of soft-computing. The choice of textbooks, papers and reports is almost numberless. Therefore it is important to choose the most relevant publications supplemented by articles about novel ideas and developments.

To promote the ideas of this research a number of publications were made and presentations at international conferences given.

1.3.1 Anticipated Contributions

The anticipated contributions of this research work are a software system that incorporates:

- ▷ a new approach for complex image processing and artificial vision tasks,
- ▷ a combination of statistics, image processing and soft-computing,
- ▷ usage of high order statistics,
- ▷ a possibility for the optimisation of the system,
- ▷ and a solution to overcome the problems arising from existing texture collections.

1.4 Introduction to the Work

In this work all those techniques are being presented that are needed to construct a complex image processing system.

The first section (2.2) in Chapter 2 introduces the solutions nature provides, i. e. the visual systems found in the fauna and especially that of the human. Even though so far not every detail of the human visual system can be fully explained by science, the system layout provides information on how an artificial representation can be designed. At the

same time it becomes evident that a one-to-one copy is not possible, as the computer hardware available today does not come close to the performance of a human brain.

Next to how the perception functions it is discussed in Section 2.3 what is being perceived. In this work this is being reduced to the recognition of two-dimensional textures. Therefore an introduction is given into what texture is. Additionally new texture subgroups are being introduced. This will give, not only for this work, a better understanding of textures and the performance of texture classifiers.

The next sections discuss in detail all the methods and techniques needed and used for the Texture Classification System. It starts in Section 2.4 with a variety of image filters and transformation methods. All the methods described in that section are used to reduce unnecessary image contents and thereby enhance the relevant information for the following steps. Examples are provided to show the effects of the described methods. As so far it is not known which methods are best suited for which kind of texture or texture class, all the approaches are treated equally. Some of the techniques are used in a different manner than for what they were originally designed. Explanations why this makes sense for texture classification are given.

The next section (2.5) describes the methods used for the extraction of texture features by the use of statistical methods. Explanations are given why low-level statistics are not sufficient for texture classification. The higher moments of the first order statistics and the second order statistics are being introduced. Examples for texture classification are given and a number of new approaches for cooccurrence matrices are presented.

At this point the system has produced a feature vector which represents a certain texture. Fig. 107 shows that the feature vector can become very large. The next step in the system is to classify this vector. The next two sections in that chapter describe classification methods.

One possible approach to classify data is by clustering (Sec. 2.6). Every classifiable object builds a cluster in the feature space. This space has the dimension equal to the size of the feature vector. Next to sharp clustering some fuzzy clustering techniques are being discussed. Due to the nature of textures and dependent on the techniques used for the construction of the feature vector overlapping clusters will be almost unavoidable. Fuzzy clustering provides a solution for this problem.

An alternative to clustering methods are neural networks (Sec. 2.7). An introduction into the technique and the capabilities is given. A great variety of neural networks exists

which are being described in literature. The variants that look most promising for this classification task are being presented.

The final section gives an introduction into genetic algorithms (Sec. 2.8). The functionality and variants of such an algorithm are explained and the details needed for the Texture Classification System discussed. The use of a genetic algorithm provides a good optimisation module for the system. The sketch in Fig. 63 shows the principle.

The Chapter 3 describes the actual design of the Texture Classification System. It starts with the Core-System (Sec. 3.2), the centrepiece of the Texture Classification System. This part performs the actual texture classification. This is followed by an introduction to the Enhanced-System (Sec. 3.3), which provides pre-processed input for the Core-System and handles its output.

The modules for the automatic optimisation are being introduced in Section 3.4. The main focus is the reduction of feature vector size. Other optimisable areas of the system are also being discussed.

A special module which is part of the system's control structure is the border detection routine 3.5. This routine enables the system to control the input and output in case of multi-textural images.

The final section in that chapter is describing the implementation of the Texture Classification System in the WiT[2] environment.

The following Chapter 4 discusses the use of the Texture Classification System. For the reasons outlined above a number of different real world applications were employed to verify the design principles and methods as far as possible with today's computer hardware. A system for quality control of glass containers (Sec. 4.2), one for quality control in veneer (Sec. 4.3), one for a positioning control of steel blocks in a rotating oven (Sec. 4.4), and a final system for the measurement of hair-gloss for product tests (Sec. 4.5) are being presented. The requirements and results of those systems are discussed.

The Chapter 5 describes by means of an example in detail the functionality of the Texture Classification System. The system's complexity and the resulting need for vast computational resources is being pictured. The need for optimisation becomes evident and the possibilities of the Automatic-Optimisation-System (AOS) are being described. This is followed by a discussion of necessary future developments.

[2]An introduction to the WiT programme is given in App. C.

The appendix gives an introduction to texture image libraries (App. A) and introduces for the first time the Stolpmann texture collection (App. A.3). This is followed by a selection of images used for the projects described in Chap. 4. Next comes an introduction to the image processing tool WiT which uses graphical programming. This again is followed by an introduction to neural network software. The possibilities of such tools are being explained. A short introduction into the author's publications is followed by a glossary, a list of abbreviations, the references and an index.

1.5 Conclusions

The holistic philosophy of the modular self-optimising design and bringing together expert knowledge of different research areas in one complex system is the main achievement of this research work.

A texture classification system has been designed for which:

- natural vision was used as a "blue-print",
- a more detailed definition of what texture is was made,
- a close review of image processing techniques provided a variety of applicable methods,
- review and enhancement of statistical methods,
- neural networks and fuzzy clustering were used for classification, while genetic algorithms provide a means for self optimisation, and
- a modular system layout was used.

The idea is not to have a specialised system with excellent benchmark results for a certain, well defined set of texture images. The Texture Classification System can be used to obtain a lean purpose build system for a special task. This new sub-system has than the potential of being an optimal system.

In future with increasing computational power it will be possible to use more and more parts, i. e. special modules, at one time under real-time conditions. To archive artificial vision it is important not to make the computing routines work faster, but to

use a great variety of methods at once, with every method performing its specialised task to generate a powerful system.

The Texture Classification System is the first instance of such a system.

1.6 Expected Novelty

This research project brings forth a number of new ideas. The mayor achievement lies in the design principle of the Texture Classification System. Especially the modular combination of statistics, image processing and soft-computing methods, easily adaptable to new tasks, and the self-optimisation to achieve lean sub-systems is a step forward in the field of image processing and artificial vision.

A number of modules themselves provide new solutions that have not been described before. The new ideas for high order statistics can be used in different applications, too.

Last but not least a new texture image library has been established. This will overcome restrictions of former collections.

Chapter 2

Vision, Texture and Methods

2.1 Objectives

In the first section of this chapter a detailed introduction into the human visual system with respect towards an artificial equivalent is given. This is followed in the second section with a description of the properties of texture. The following sections introduce all the methods and techniques that were tested and implemented for the Texture Classification System.

The human visual system is a complex image processing and classification machine. The objective is to understand the complexity of the eye-brain-system and to draw conclusions for an artificial vision system and the proposed Texture Classification System. It has to be taken into account that the functionality of the human brain is not yet fully understood and is still a field of extensive research. Additionally the capabilities of today's computers do not meet those of the human brain. Therefore the goal is to understand and re-use the principles.

So far the word *texture* is ill-defined. The objective is to provide a better and more detailed definition. Additionally a number of examples will be provided to give a better understanding of the subject.

The third section explores image processing techniques, mainly filters and transformations. The objective is to test existing methods with texture images as input. During those tests it is important to be open minded for new approaches. It is possible to use some methods in a different way, especially in conjunction with the statistical methods described in the forth section.

Very soon it became clear that the normal approach of using statistics for image processing as described in literature is not sufficient for texture classification. New approaches have to be designed and tested, especially statistics of higher order.

The following two sections describe alternative methods used for classification. Firstly this is clustering and fuzzy clustering, secondly neural networks in general and some special networks that are supposedly suitable for texture classification.

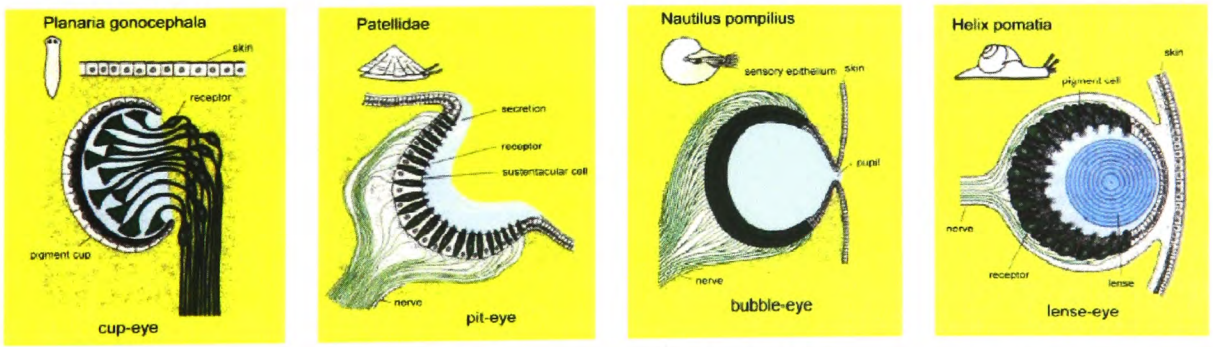
The last section in this chapter looks into system optimisation. At this point it is clear that the Texture Classification System will become fairly complex, thus an automated optimisation is needed for speed-up purposes. From all possible solutions using genetic algorithms seems to be the most promising method. The functionality and capabilities of genetic algorithms are described in this section.

2.2 The Human Visual System

2.2.1 The Eye

The eye *(lat. oculus)* is a photosensitive organ of animals and humans. Light intensities are being detected by chemical reactions in the pigment cells. Thus information about the surrounding world is being obtained.

2.2.1.1 Types of Eyes



(a) Cup-eye (b) Pit-eye (c) Bubble-eye (d) Lens-eye

Figure 1: Illustrations of eyes of simpler life-forms. (adapted from [vob])

The most simple light sensing organs are stigmas of unicellular organisms, which are plasma areas containing carotenoids. On the next step of evolution a photosensitive substance in a vacuole lies under the translucent skin. Earthworms are equipped with such cells. Both kinds can only detect light.

The pigmented cup-eye (Fig. 1(a)) of some worms and snails allows direction-dependent seeing. Only light from the off-side of the pigmented cup will be detected.

If the pigmented cells are on the ground or sides of an indented part of the skin, it is called pit-eye (Fig. 1(b)). Most snails have such eyes. If the opening is small it is a camera-eye or hole-eye, working just like a camera-obscura. If that hole is filed with a secretion it is a bubble-eye (Fig. 1(c)). The nautilus has such eyes. The quality of the sensory information is increased, but light-sensitivity is rather low.

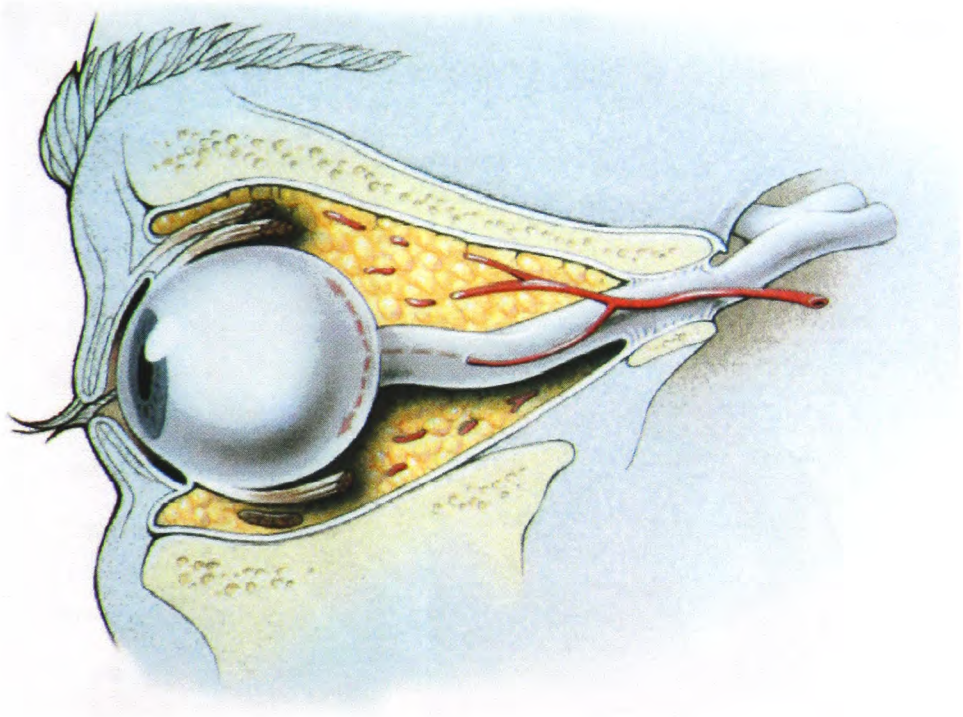


Figure 2: Cut through the human eye socket. (from [cdp])

If the hole is replaced by a lens it is called lens-eye (Fig. 1(d)). All vertebrates and cephalopods have those eyes.

On a side-branch towards a powerful eye insects have developed facetted eyes (compound eyes). An eye consists of hundreds of simpler eyes, called ommatides.

If an eye has different kinds of pigmented cells (photo receptors) it is possible to differentiate between colours. Binocular seeing is possible if the field of vision of two eyes overlap. The brain can than measure the distance towards an object. [CLZ81]

2.2.1.2 The Human Eye

The human eye is a complex lens-eye (Fig. 2). The light sensitive sensory organs are located in the retina. Each eye consists of approximately 6-7 million colour sensitive sensors , so called cones. They are concentrated around the fovea (yellow spot), the centre of the retina. The cones can again be divided into three groups of different spectral perception: red, green and blue. Additionally there are about 120 million rods, which detect brightness. They are more evenly distributed over the retina with the exception of the area around the yellow spot (Fig. 3). Numerous rods are connected to only one nerve, 130 on average, to enhance the sensitivity. (cf. Fig. 4) [Abm94, Ste93b]

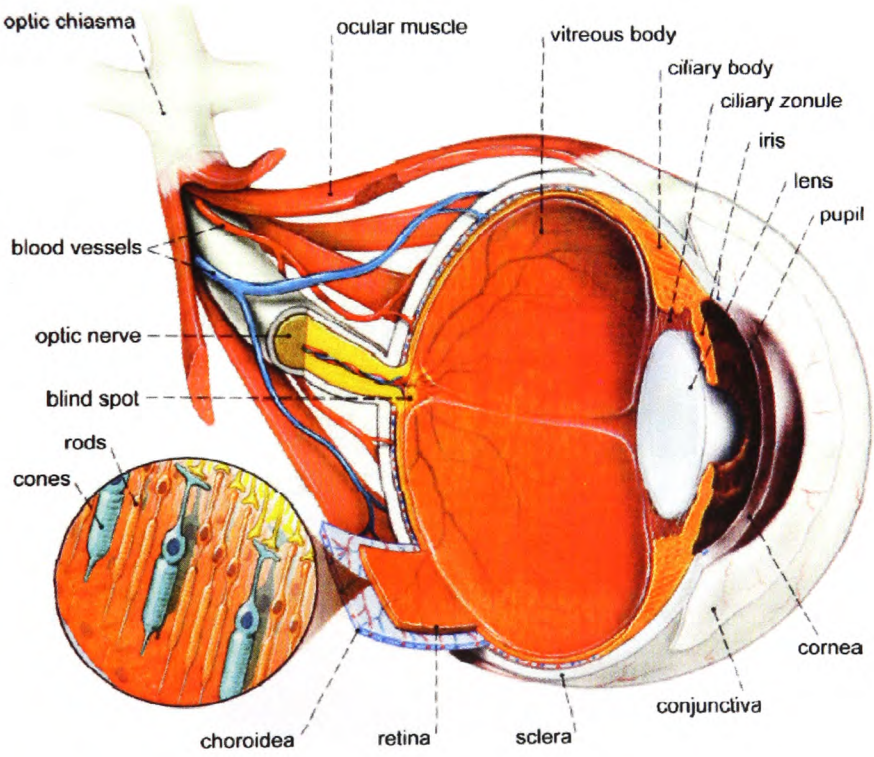


Figure 3: The eye and optic nerve. (adapted from [Wei02])

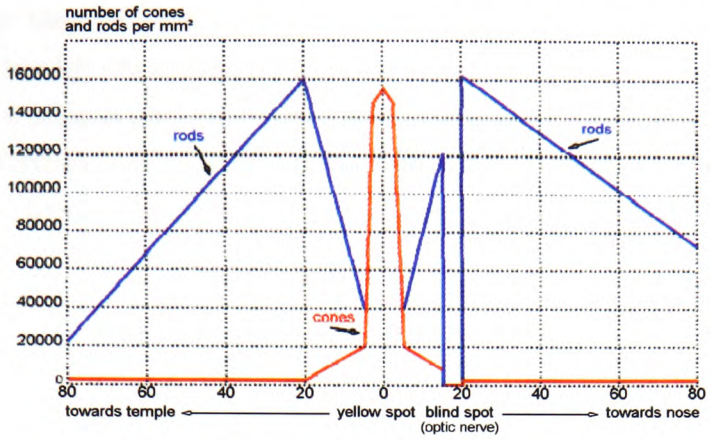


Figure 4: The distribution of cones and rods on the retina of the human eye. (adapted from [Ste93a])

The light that falls onto the retina starts a chemical reaction which is transformed into electric impulses by the photo receptors, the first neural layer. The impulses are being processed through the bi-polar ganglion cells, the second neural layer, and the large ganglion cells, the third neural layer (cf. Fig. 5). The axons of the third neural layer are connected to the optic nerve.

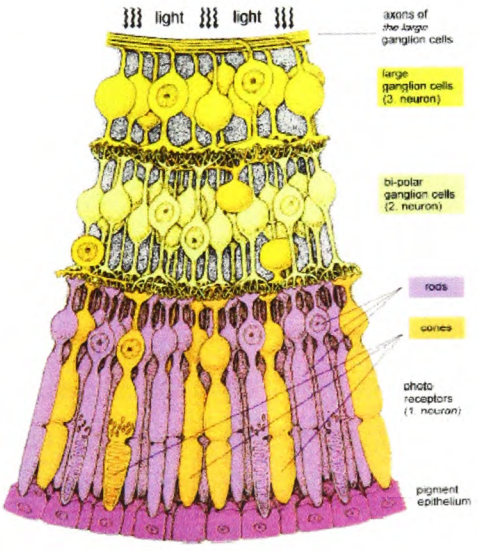


Figure 5: The retina of a human eye. (adapted from [SM02])

The reaction time of rods is about 80-90 ms, cones react within 300 ms [Ste93b]. The signals are locally averaged and about one-million nerves per eye, the optic nerve, transport the information into the brain. Further filtering and information conglomeration routines are performed in the optic chiasma, the lateral geniculate body and the visual cortex, where the image representation takes place.

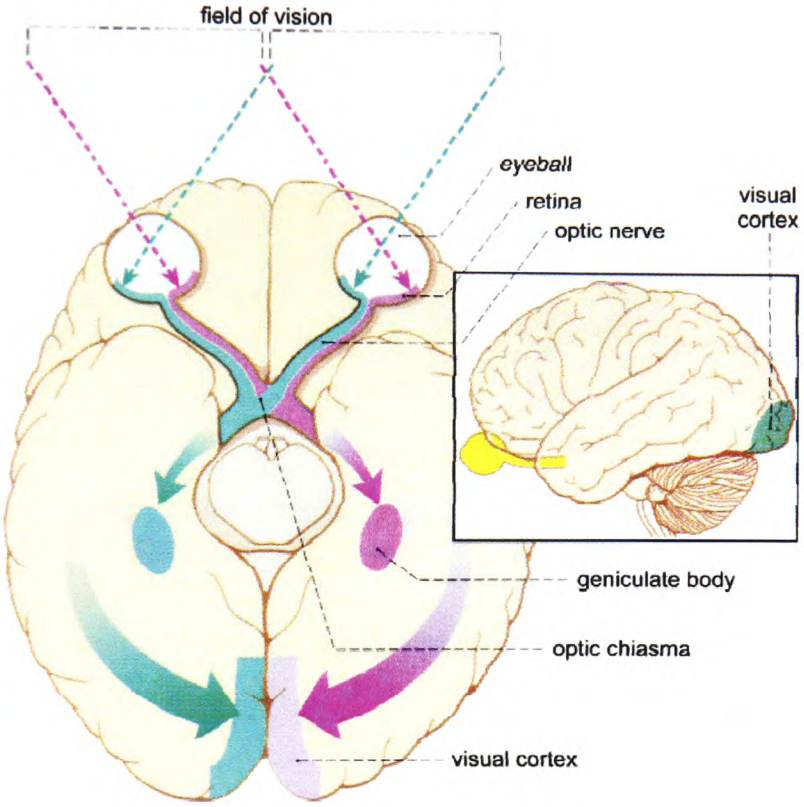


Figure 6: The human visual system. (adapted from [Wei02])

The angle of stereoscopic vision is about 30 degree with only 1 degree around the optical axis being focused. Towards the fringe area movement is only recognised indistinctly. Horizontal and vertical structures are detected about 10% better than diagonal ones due to specialised neurons in the brain [CM87].

The human eye can distinguish between about 160 different colours. Are greylevels, hue and saturation taken into account about 7 million colour shades can be recognised [Bös95]. In black and white images only 30 different greylevels are detected [Nie82].

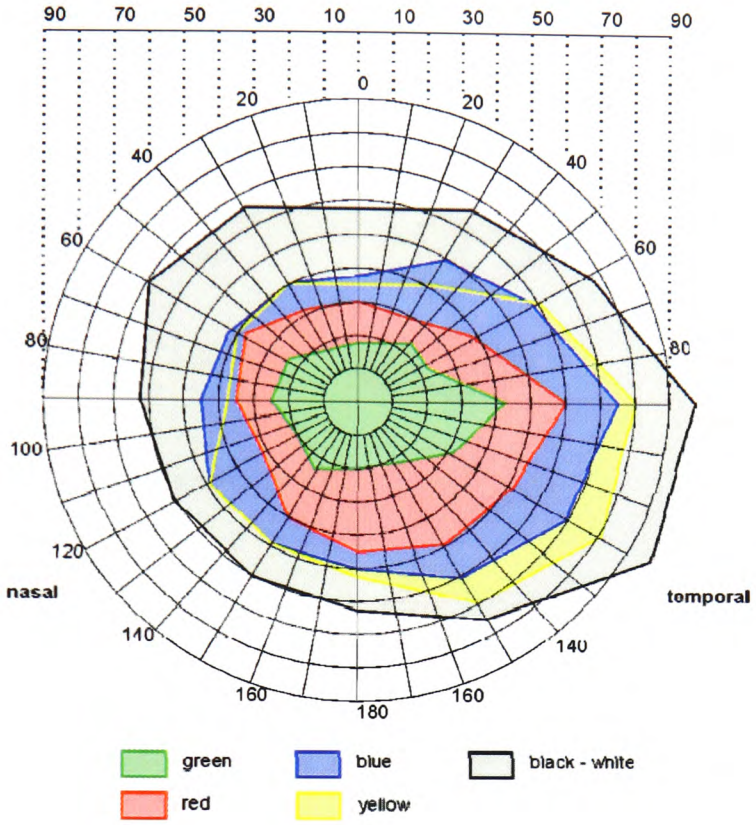


Figure 7: Colour Vision. (adapted from [Roc])

The information extraction is not performed in one place but the signals are processed through several steps and more and more abstracted. This shows that for an artificial system module components are preferable to one-pass solutions.

2.2.2 The Brain

Just like the eyes became more and more complex by evolutionary development, even more so did the brain. Whilst the eyes of vertebrates are similar between the genii, brains differ quite considerably (cf. Fig. 8). The cerebrum grew along with the capabilities of the visual system.

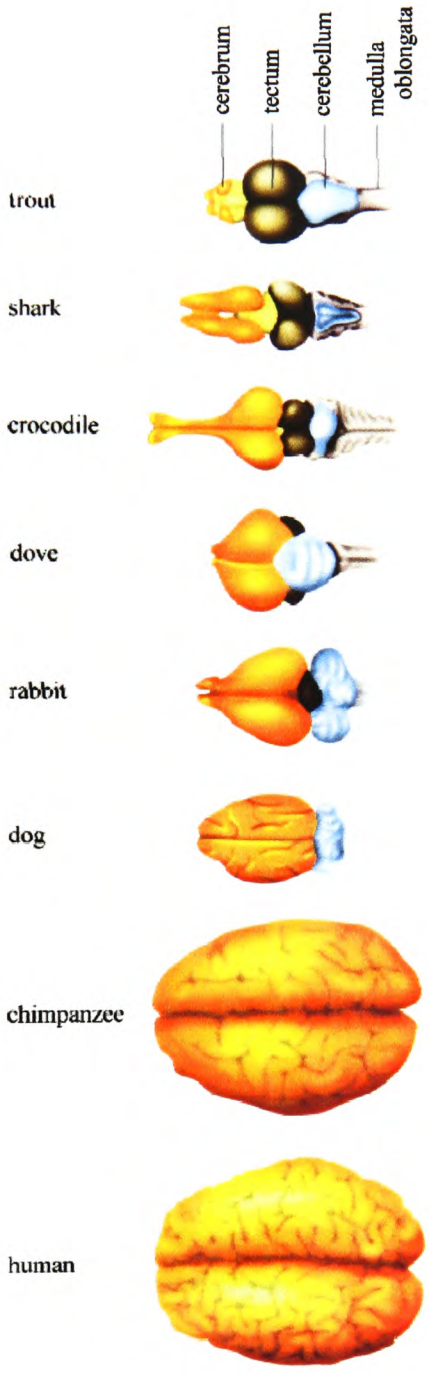


Figure 8: Sizes of the cerebrum of vertebrates. (adapted from [Geh])

To be able to fulfil the complex visual tasks that are important for every-day life, the optic information is processed from the sensors (retina) through the optic nerve to the optic chiasma. The optic chiasma is important for the binocular (or stereoscopic) vision. The information is than processed along the optic tract to the geniculate body, and from there along the optic radiation to the visual cortex. (cf. Figs. 6, 9)

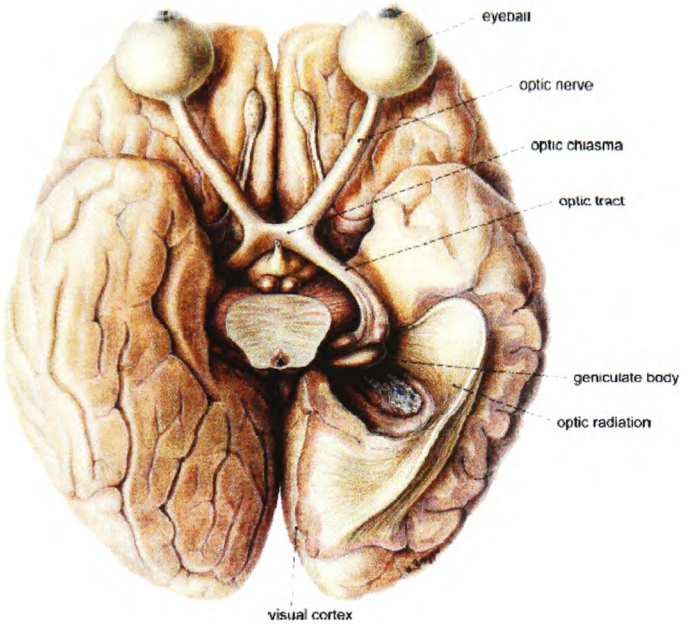


Figure 9: The optic chiasma. (adapted from [Wei02])

About 100 million simple pieces of information reach the eyes every second. These are reduced by a rate of one million to one to about 100 complex pieces of information. With this complex information the brain builds a representation of the surrounding world. This is what our brain tells us, what we see.

Remembering that only 1 degree around the optical axis gives a sharp images and that colour vision takes only place in a small part of the retina (cf. Fig. 7), it is due to the capabilities of the brain that we think we see everything sharp and in colour.

That means that artificial vision systems still have a long way to go before they will reach or better the human visual system. Especially the human 'processing unit', the brain, is more powerful than computers are today. (See also Section 2.7)

2.3 Texture

One of the problems of artificial vision is the detection and classification of coarse-grained non-regular structures. Such structures are best described as texture.

As the system designed in this project has the aim of detecting and classifying texture, it is necessarily to define what texture is and what kinds of textures exist.

2.3.1 General Description

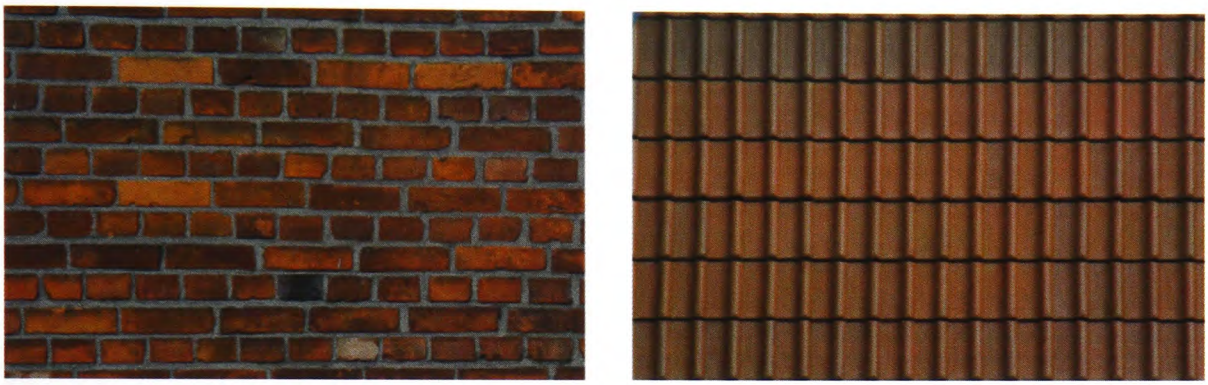
There is no unique definition of the word *texture*, but rather a context dependent description. It descends from the Latin word *texere* ("to weave") and the WEBSTER [Web93] definition closest to the use in image analysis is *similar qualities dependent on the nature and arrangement of the constituent particles of a substance.*

The word texture is linked to a number of different subjects. To better understand the meaning and usage of the word, for which no clear definition exists, examples of the usages different to image processing are being given in the following list [Wik05]:

- In music, the word texture is often used in a rather vague way in reference to the overall sound of a piece of music. A piece may be described as having a "thick" texture, or a "light" texture, or other terms taken from outside of music. The perceived texture of a piece can be affected by the number of parts playing at once, the timbre of the instruments playing these parts and the harmony and rhythms used, among other things.
- In cuisine, texture is the feel of food on the tongue and against the teeth. Adjectives include crunchy, soft, sticky, mushy, hard, spongy, and airy.
- In painting, texture is the feel of the canvas based on the paint used and its method of application.
- In materials science, texture is the property of a material's individual crystallites sharing some degree of orientation. It is seen in almost all engineered materials, and has a great influence on material properties.
- In computer graphics, a texture is a bitmap image used to apply a design onto the surface of a 3D computer model.

In real-world images a texture might be best described as a non-regular structure consistent of elements that hold a certain likeliness among one another. Those elements are the smallest identifiable part in a textured area and are often being referred to as texels. The following sections give examples and more detailed explanations.

2.3.2 Types of Texture



(a) Brick wall (b) Rooftiles

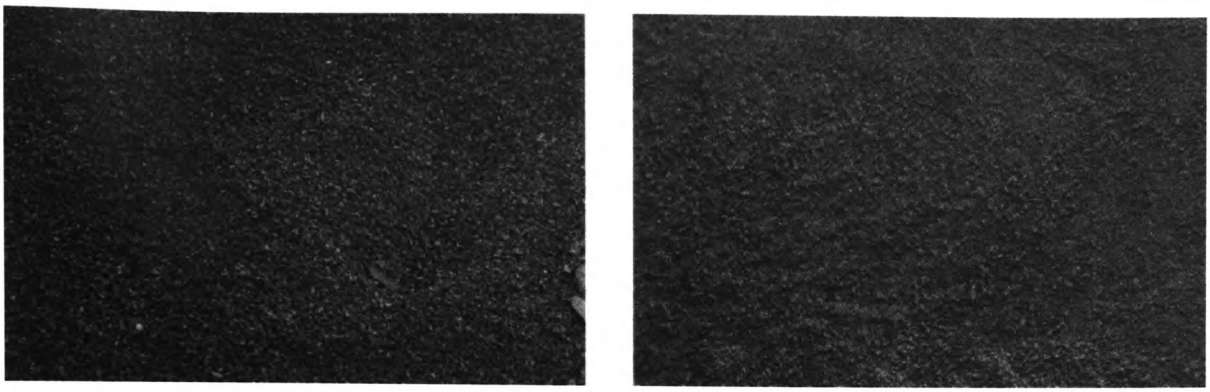
Figure 10: Regular texture.

In general it can be said that a texture describes the composition of an object[1]. It is useful to divide texture into regular and statistical texture. Regular texture is composed of repeated texture primitives, also called texels, which are large against the pixel resolution and could be described in further detail (cf. Figs. 10(a), 10(b)), whereas the elements of a statistical texture near unity and have a random distribution (cf. Figs. 11(a), 11(b)). Very often texture is of a hierarchical kind where the macro texture is regular and the micro texture of the primitives statistically describable [BB82, SHB93, Ste93b, Wer85]. A good example for this are the images in Figs. 12(a), 12(b).

2.3.2.1 Artificial and Genuine Texture

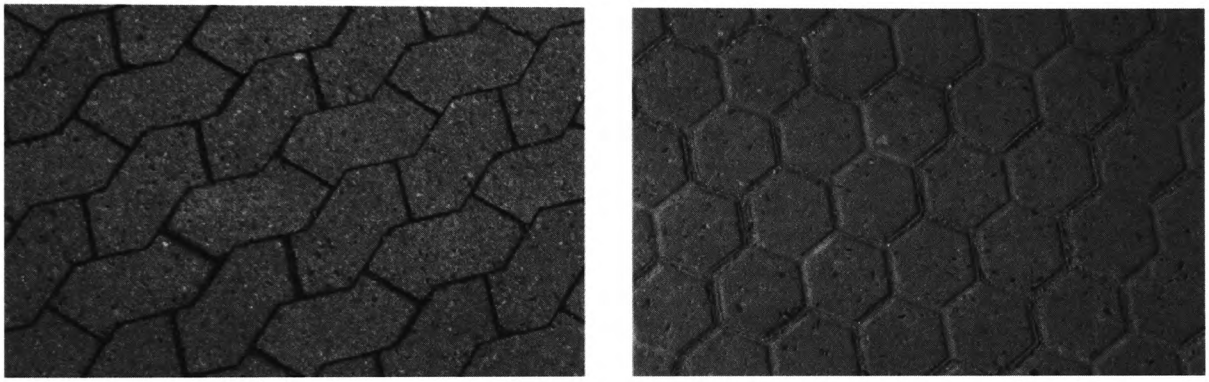
Two major groups of texture types exist, namely artificial and genuine texture. These texture types are not to be confused with the texture definitions as described above. Here again it can be observed that the word "texture" is ill defined and context dependent.

[1]In this project only the surface composition or any projection onto a 2D-image-plane shall be examined



(a) Grit (b) Roughcast

Figure 11: Statistical texture.



(a) Interlocking pavement (b) Hexagonal pavement

Figure 12: Hierarchical texture.

Artificial texture is usually computer generated and thus well defined. Such texture was widely used at the end of the last decade as web-page backgrounds. As this was rather more distracting than helping the reader such textured backgrounds became unfashionable again. But in the field of computer graphics artificial textures are being used to cover surfaces of 3D-models, as calculating e. g. skin, fur or cloth could not be done in real-time, but mapping a texture onto an even-surfaced model in real-time is possible. Handling this kind of texture in a texture classification system may result in finding parameters that describe the regularity instead of the texture itself.

(a) Small flowers

(b) Dried blades of grass

(c) Gravel

(d) Old Bark

Figure 13: Natural texture.

Genuine texture on the other hand can only be found in our surroundings. To handle these non-virtual textures photographs have to be taken and subsequently transferred into a computer readable format. After that it is possible to use genuine textures for further investigation.

The genuine texture itself can be subdivided into three groups, natural, half-natural and man-made texture. Textures cannot be assigned strictly but rather in a fuzzy way to a certain group.

In this project only genuine textures are being considered.[2]

[2]It is of course possible to handle artificial texture but no concern is given towards its special features that separate it from genuine texture.

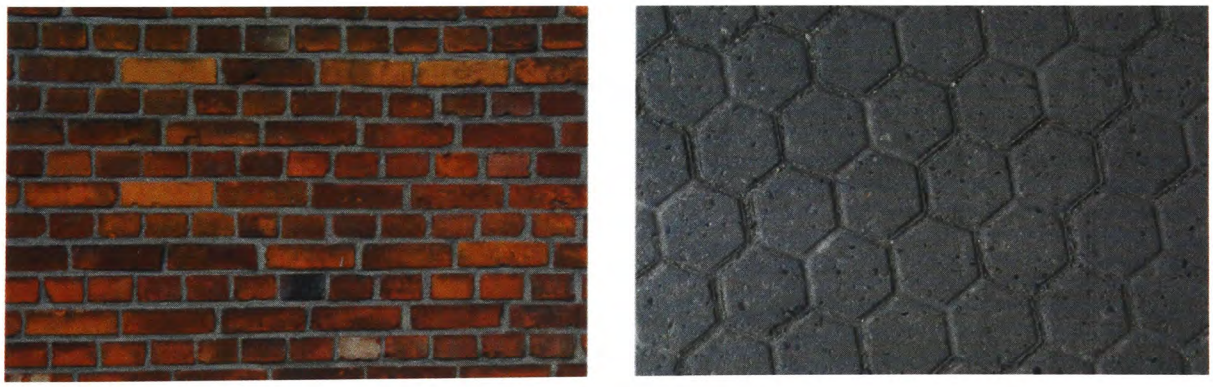
2.3.2.2 Natural, Half-Natural, and Man-Made Texture

The author will divide the group of genuine textures into three subgroups — natural, half-natural, and man-made texture. This subdividing of texture is a new idea of the author and has been introduced for the sake of being able to measure statistical differences between those groups. This information can be used to tell natural from man-made texture even if the texture itself is not known to the classification system.

Natural Texture

Natural textures are those that can be found in unspoilt nature, that means in areas untouched or unchanged by mankind. One has to be aware that finding such places can be difficult. Humans have put their mark on almost everything within their reach. For example in Europe almost all forests are being harvested and trees are replanted in straight lines. There is no meadow that has not seen agricultural machinery. As taking pictures of the treetops of the Amazonian forest or sand dunes in the Sahara desert is not very helpful, one has to look at the small scale textures. The pictures in figures 13(a) to 13(d) show some examples.

Man-Made Texture



(a) Brick wall

(b) Hexagonal pavement

Figure 14: Structured man-made texture.

In contrast to the natural textures some man-made textures are well structured. Even well defined edges that describe texture elements can be found. The next group of pictures in figures 14(a) to 14(b) give a good example.

But not all man-made textures are so well defined. The next pictures show some examples that have no or only unstructured edges. Those textures belong to the class of statistical textures. (cf. Figs. 15(a), 15(b))

(a) Old pavement (b) Old stonewall

Figure 15: Unstructured man-made texture.

Half-Natural Texture

The third group of textures, half-natural textures, are textures that are natural but formed by man. Examples are a lawn, a cornfield, a hedgerow and even a wall made of concrete. If one looks at the picture 16(a) in detail one will see the structure of wooden planks and pebble stones embedded in the concrete. Wood, even though a natural substance, shows, if cut up, an unnatural texture.(cf. Figs. 16(b))

2.3.3 Multi-textural Images

The real world does not consist of plain textures but is a composition of larger or smaller fragments of texture. The boundaries between different kinds of texture are sometimes fuzzy, sometimes sharp, sometimes in a straight line and sometimes curved or fractal.

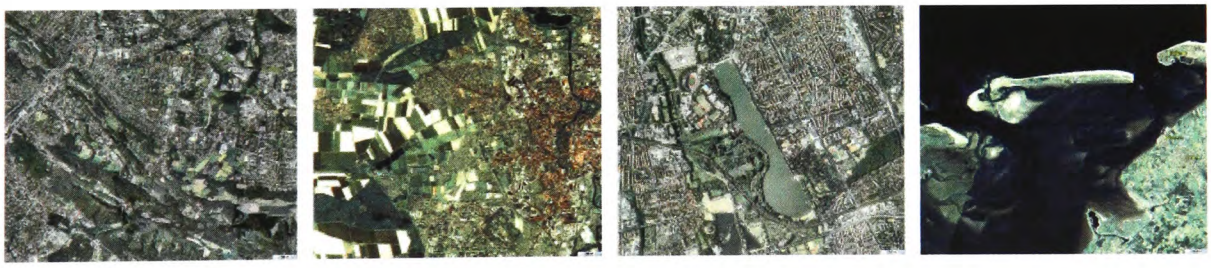
The following pictures show examples that point out the difficulties a visual recognition system has to handle.

(a) Concrete (b) Wooden fence

Figure 16: Half-natural texture.

2.3.3.1 Sharp Borders

The satellite photographies in Figs. 17(a) to 17(d) show examples of multi-textural images with sharp boundaries between single textured areas. The definition of which part in such images is texture, and which is an object, depends on the scale being used for texture detection and on the detail resolution of the image. The houses in Fig. 17(c) are on a large scale a regular texture (cf. definition in Section 2.3.2). Zooming in on the image will reveal different textures like roofs, streets, courtyards, gardens, etc.

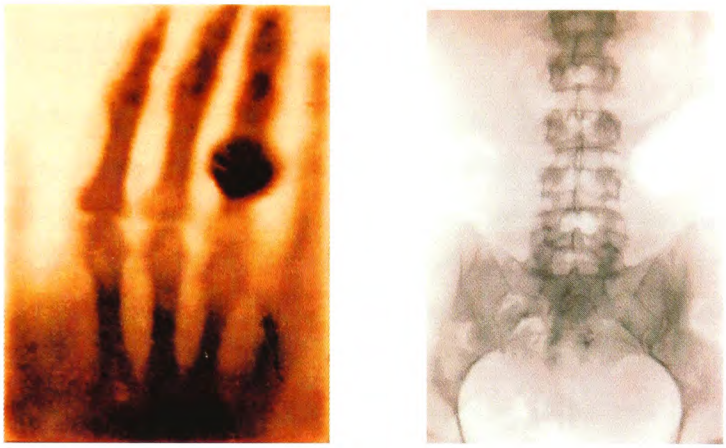


(a) Bielefeld, centre and Teutoburger Wald (b) Braunschweig, West and centre (c) Hannover, centre and Maschsee (d) Juist with tidal shallows

Figure 17: Satellite images.

2.3.3.2 Fuzzy Borders

Very often there are no sharp boundaries between textures, so called fuzzy-bordered multi-textural images. The x-ray images in Figs. 18(a) and 18(b) give a good example. Also the tidal shallows in the satellite image 17(d) show fuzzy-bordered areas. Such areas could be identified using a texture-threshold or by giving a membership value for several textures. A way how to detect fuzzy borders with the help of the Border Detection Routine (BDR) designed by the author is described in Section 3.5.2. So far no system is known to the author that is comparable to the BDR.



(a) The hand of Mrs Wilhelm Röntgen: the first x-ray image, 1895 (from [Gla33])

(b) Modern x-ray image of lower spinal cord (from [UWL])

Figure 18: X-ray images.

To obtain a system that describes texture properties within a picture or scene it is necessary to handle sharp and fuzzy bordered multi-textural images. A special border detection routine has been designed for that task (see Section 3.5).

2.4 Image Filtering and Transformation

A great number of image preprocessing methods already exist and are being used in contemporary image processing systems (cf. [Abm94, BB82, BB93, Ern91, Hab91, Ric93,

SHB93, Ste93b]). A variety of these shall be used to reduce the amount of unnecessary data or to enhance special properties within the image.

It is possible to put any kind of method to use within the Texture Classification System and a large number of such methods have been tested. Chapter 3 describes what methods have been used in which way. A detailed description of their possibilities and potentials can be found in literature. Some examples and new ideas like wavelets, LAWS micro structure texture measures, as well as more widely known filtering and transformation methods will be described in this chapter.

2.4.1 From Image to Image

Unlike most approaches towards identifying textures, colour images will be used in the Texture Classification System. This has the benefit of being able to use the colour information within an image. At the same time it makes the processing routines more complicated. Most filtering techniques or transformation methods are designed to work on monochrome images. The digitised colour images themselves are in fact three[3] layers of monochrome images, each representing one primary colour, usually red, green and blue.

For the use in the Texture Classification System the 24-bit RGB colour image is being split up, either its red, green and blue channels, or be changing the colour space first.

24-bit colour images can be converted from the CIE spectral primary system (RGB) colour space into the hue, saturation, value (HSV) colour space or into the CIE uniform chromaticity scale (Yuv) colour space. The Y channel corresponds to the image luminance and generally provides a good grey scale representation of the colour image. The u and v channels encode the chrominance or colour values of the image. In Yuv image compression the Y channel is often given greater precision than the u and v channels since human vision is more sensitive to brightness changes than to colour changes.

Most of the images shown in this chapter were reduced to the value and to a size of 1080×720 pixel before being filtered or transformed.

[3]Satellite images have up to seven colour planes, e. g. special ones for infra-red (IR) or near-infra-red (NIR) [Ric93]

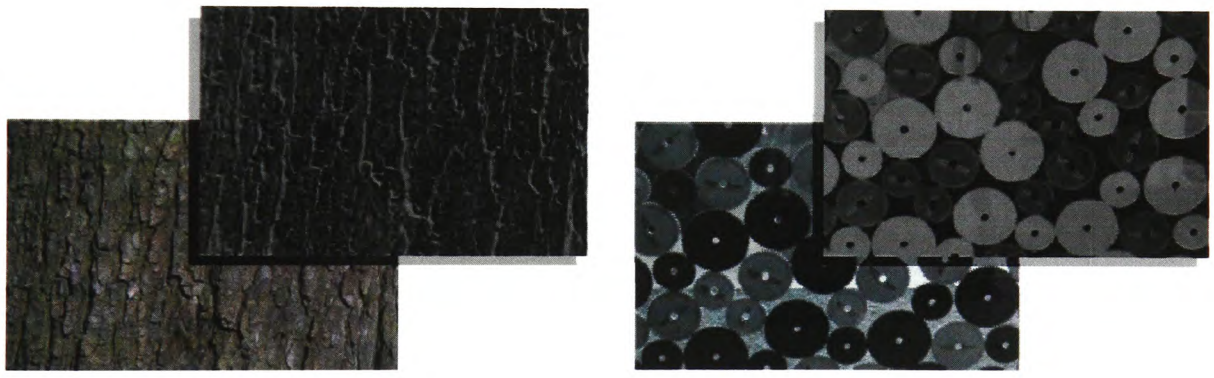
2.4.2 General Two-Dimensional Convolution

A user specified kernel is being applied for a two-dimensional convolution of the input image. With an appropriate choice of kernel values, it can perform such operations as smoothing, edge or peak enhancement, position shifting on the image. The two dimensional convolution kernel is an integer array with user specified size and weight values.

Fig. 19 shows some examples where the convolution kernel shown in Table 1 was used.

-24	1	-24
1	24	1
-24	1	-24

Table 1: Special convolution kernel.



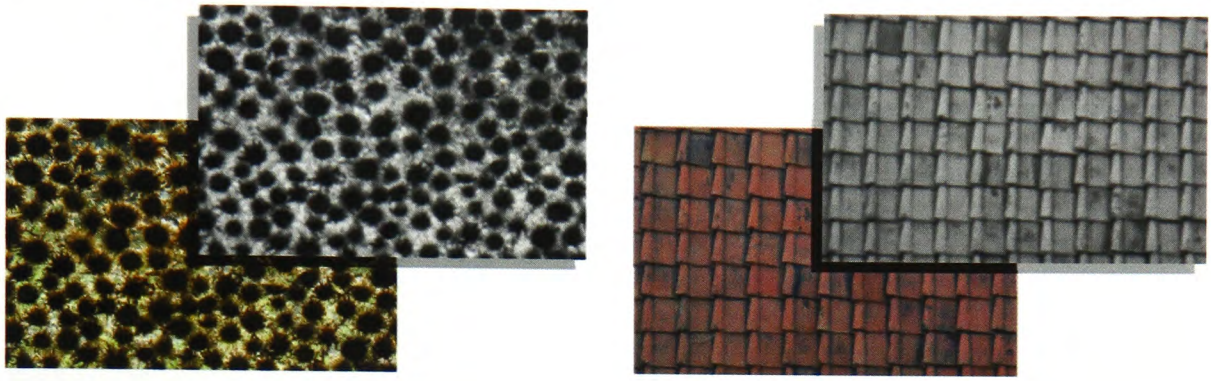
(a) Bark (b) Gearwheels

Figure 19: Convolution with a 3 × 3 kernel.

2.4.3 Two-Dimensional Smoothing – the Low-Pass Filter

The two-dimensional low-pass filtering, or smoothing, has the effect of blurring the image. It reduces the sharpness of edges and fine detail in the image and reduces the effects of high frequency noise. Both of these effects may be useful before segmentation or feature

recognition is performed. The smoothing operation is performed with a moving average kernel of a pre-defined size. The effect is equivalent to that of convolution with a kernel of the same size and all kernel values equal to 1, and a scale factor equal to the number of elements in the kernel array.[4]



(a) Dried Flowers (b) Rooftiles

Figure 20: Low pass filtering with a 9×9 kernel.

The images in Fig. 20 were obtained by using a 9×9 filter as shown in Table 2 and a scale factor of 81.

1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

Table 2: Low-pass kernel of the size 9×9 for a scale factor of 81.

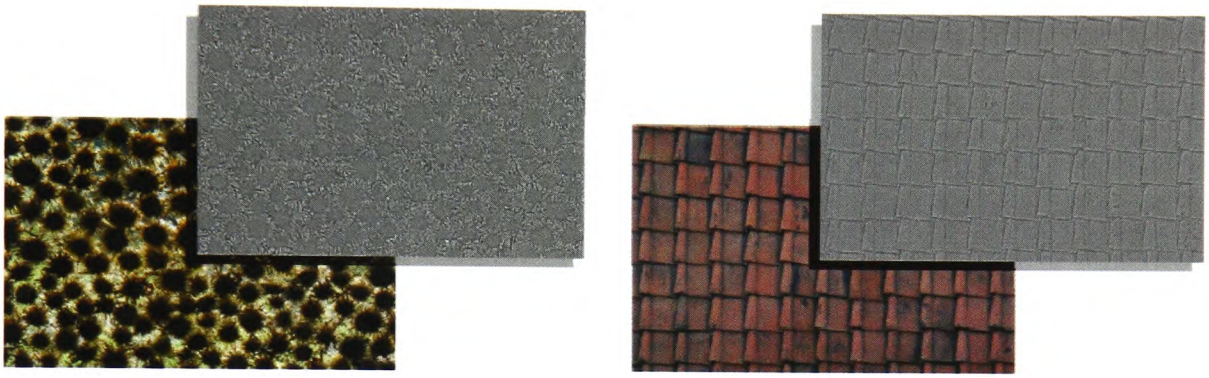
2.4.4 Two-Dimensional High Frequency Enhancing – the High-Pass Filter

The two-dimensional high-pass filtering, or sharpening, enhances local contrast and suppresses slow changing values. This technique is helpful in enhancing image detail and

[4]The simple kernel structure allows a more efficient implementation.

edges and in eliminating gradually changing global effects such as lighting variations. First, a smoothed image is calculated over the image as described in Section 2.4.3. The smoothed image is then subtracted from the original image to give the high pass filtered image. Increasing the kernel size lowers the cut-off frequency of the filter and decreases local contrast enhancement.

The images in Fig. 21 were obtained by using the same 9×9 (low-pass) filter and scale factor as for the images in Fig. 20.



(a) Dried Flowers (b) Rooftiles

Figure 21: High pass filtering with a 9×9 kernel.

By adding the low and high-pass filtered images the original image is being retrieved.

2.4.5 Two-Dimensional Gaussian Smoothing

For the two-dimensional gaussian smoothing of an image a discrete approximation to a gaussian kernel is being applied. The effect of the smoothing operation is to blur the image, reducing the sharpness of edges, fine detail and the effects of high frequency noise. An analytic gaussian filter more closely approximates a perfect low-pass filter than the moving average filter described in Section 2.4.3, while avoiding the ringing artefacts associated with the perfect low-pass filter.

The implementation for the Texture Classification System provides four different discrete approximations to gaussian filters. A filter of size 2×2, 3×3, 5×5 or 7×7 may be selected. Use of the gauss operator is equivalent to using a two-dimensional convolution with the kernel and scale parameters shown in Table 3.

The images in Fig. 22 were obtained by using the 7×7 kernel shown in Table 3.

1	1
1	1

2 × 2
scale 4

1	2	1
2	4	2
1	2	1

3 × 3
scale 16

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

5 × 5
scale 256

0	5	13	16	13	5	0
5	25	52	64	52	25	5
13	52	102	126	102	52	13
16	64	126	156	126	64	16
13	52	102	126	102	52	13
5	25	52	64	52	25	5
0	5	13	16	13	5	0

7 × 7
scale 2048

Table 3: Gaussian kernel types.

2.4.6 Laplacian 2nd-Order Edge Enhancement

The laplacian filter enhances changes in local contrast by applying a filter that approximates a laplacian second derivative operation to the image. The filter subtracts an average of neighbouring pixels from each pixel in the image emphasising points, lines and edges and suppressing values in regions of constant gradient.

For the Texture Classification System three types of laplacian filter kernels, corresponding to different methods of averaging the neighbouring pixels, are available.

0	-1	0
-1	4	-1
0	-1	0

Type 0

-1	-1	-1
-1	8	-1
-1	-1	-1

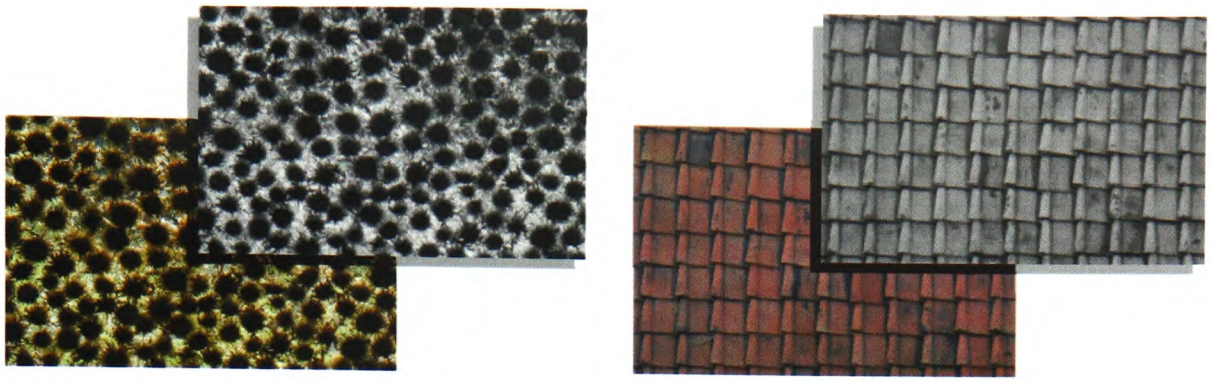
Type 1

1	-2	1
-2	4	-2
1	-2	1

Type 2

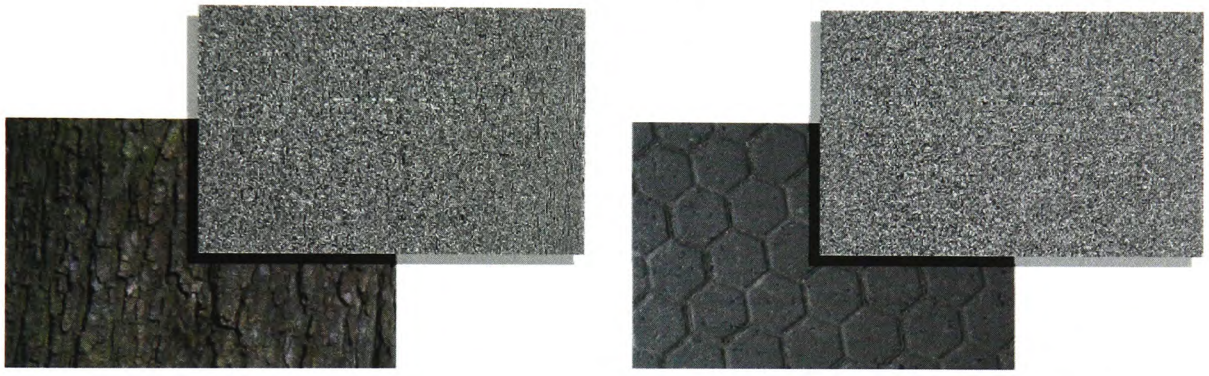
Table 4: Laplacian kernel types.

The images in Fig. 23 were obtained by using the type 1 kernel shown in Table 4.



(a) Dried Flowers (b) Rooftiles

Figure 22: Gaussian smoothing with a 7×7 kernel.



(a) Bark (b) Hexagonal pavement stones

Figure 23: Laplacian 2^{nd}-Order Edge Enhancement using type 1 filter kernel.

2.4.7 Local Adaptive Threshold

The local adaptive threshold applies thresholding to a grey scale image using thresholds based on local pixel values in the input image. The function is similar to that of fixed threshold, except that the threshold values are calculated for each pixel in the image based on pixel values in its neighbourhood.

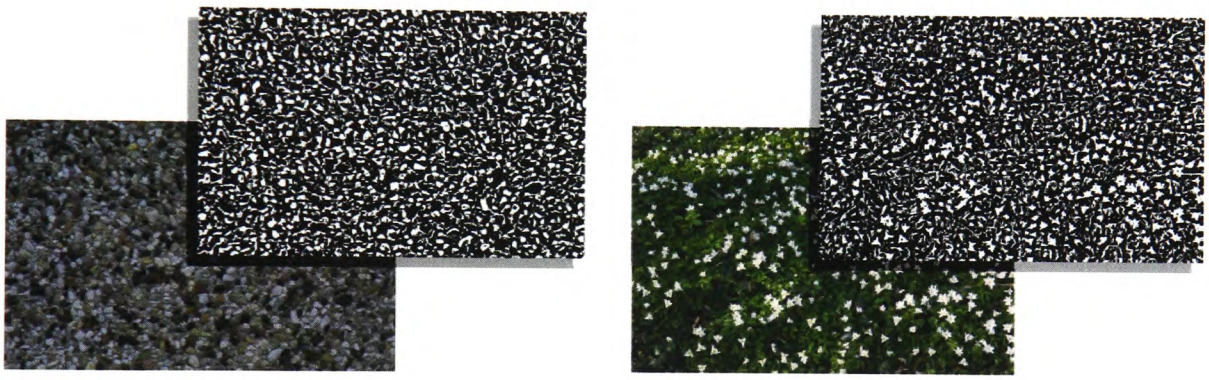
Each threshold value is set to the local mean pixel value plus an offset. The local mean is calculated over a rectangular region around the pixel whose size can be specified.

In the implementation for the Texture Classification System there is a choice of four different methods to calculate the offset.

1. An offset-value is given.
2. The difference between the local mean pixel value and the image minimum pixel value is multiplied by the given parameter value and the result is the offset value.
3. As before, but using the image maximum pixel value instead of the minimum.
4. The offset is determined by multiplying the standard deviation of the pixel values in the neighbourhood by the given parameter value.

In each case the resulting offset value is added to the local mean value.

To obtain the images in Fig. 24 an absolute offset of 10 was added to the mean over a 25×25 pixel wide region.



(a) Exposed aggregate concrete (b) Small flowers

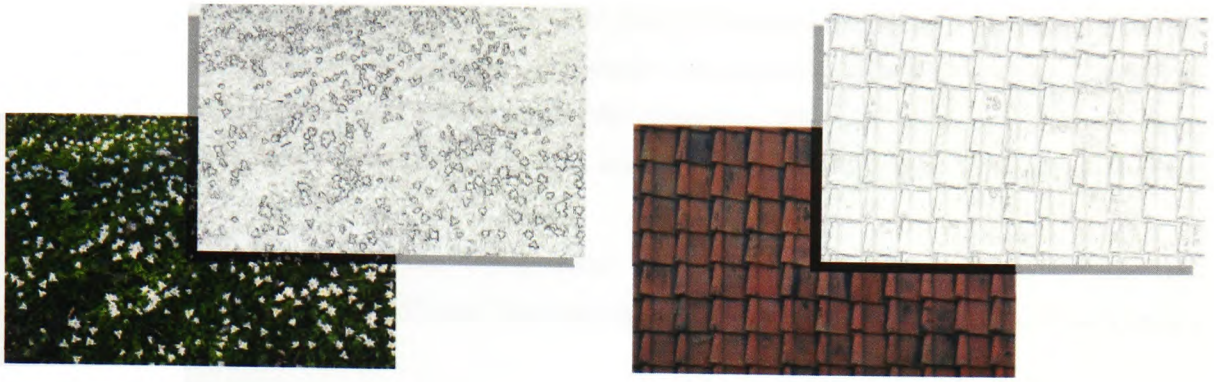
Figure 24: Local adaptive threshold.

2.4.8 SOBEL and PREWITT Gradient Filters

The gradient filter computes an approximation to the gradient of the input image. Either SOBEL or PREWITT filter weights can be used as gradient kernel.

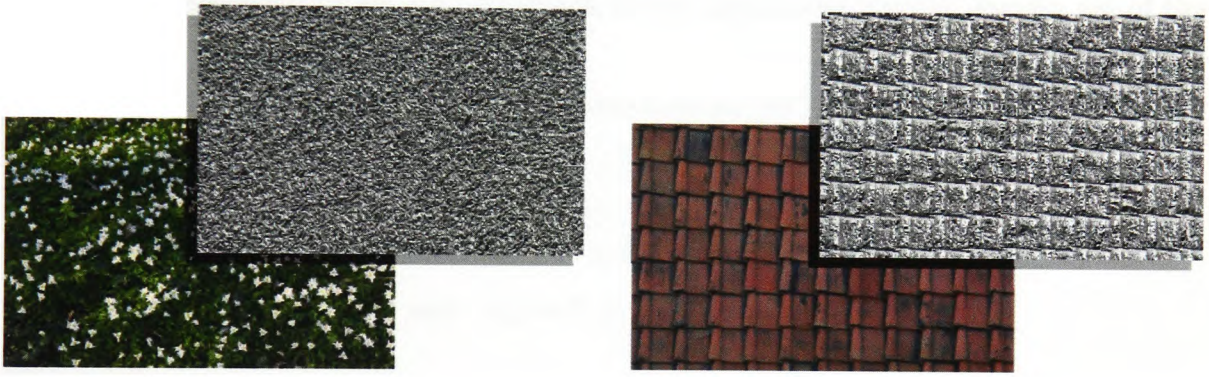
Two output images are being produced, representing the magnitude and direction of the gradient vectors at each pixel. The direction is given in degrees, in the range $-180°$ to $180°$. The gradient filter is often used for edge enhancement, with the magnitude related to the slope of the edge and the direction related to its orientation.

Figs. 25 and 26 shows examples where the PREWITT kernel was used.



(a) Small flowers (b) Rooftiles

Figure 25: Gradient magnitude filtered with PREWITT kernel.



(a) Small flowers (b) Rooftiles

Figure 26: Gradient orientation filtered with PREWITT kernel.

2.4.9 Fourier Transformation

The Fourier Transformation[5] is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

[5]Common names are Fourier Transform, Spectral Analysis or Frequency Analysis.

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression. For the Texture Classification System it can be put to use as a special frequency filter or, in a less common way, the obtained frequency domain image can be analyzed by statistical means (cf. Section 2.5).

As only digital images are being used, this section will be restricted to the Discrete Fourier Transform (DFT) and its computationally faster equivalent the Fast Fourier Transformation (FFT).

The DFT is the sampled Fourier Transform and therefore does not contain all frequencies forming an image, but only a set of samples which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, i. e. the image in the spatial and Fourier domain are of the same size.

For an image of size $M \times N$, the two-dimensional DFT is given by:

$$\begin{aligned} F(k,l) &= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-i2\pi(\frac{km}{M}+\frac{ln}{N})} \\ &\text{with} \quad m,k = 0,1,\ldots,M-1 \\ &\text{and} \quad n,l = 0,1,\ldots,N-1 \end{aligned} \tag{1}$$

where $f(m,n)$ is the image in the spatial domain and the exponential term is the basis function corresponding to each point $F(k,l)$ in the Fourier space. The equation can be interpreted as: the value of each point $F(k,l)$ is obtained by multiplying the spatial image with the corresponding base function and summing the result.

The basis functions are sine and cosine waves with increasing frequencies, i. e. $F(0,0)$ represents the DC-component of the image which corresponds to the average brightness and $F(M-1,N-1)$ represents the highest frequency.

In a similar way, the Fourier image can be re-transformed to the spatial domain. The inverse Fourier transform is given by:

$$\begin{aligned} f(m,n) &= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(k,l) e^{i2\pi(\frac{km}{M}+\frac{ln}{N})} \\ &\text{with} \quad m,k = 0,1,\ldots,M-1 \\ &\text{and} \quad n,l = 0,1,\ldots,N-1 \end{aligned} \tag{2}$$

To obtain the result for the above equations, a double sum has to be calculated for each image point. However, because the Fourier Transform is separable, it can be written as

$$F(k,l) = \frac{1}{N}\sum_{n=0}^{N-1} P(k,n)e^{-i2\pi\frac{ln}{N}} \tag{3}$$

where

$$P(k,n) = \frac{1}{M}\sum_{m=0}^{M-1} f(m,n)e^{-i2\pi\frac{km}{M}} \tag{4}$$

Using these two formulas, the spatial domain image is first transformed into an intermediate image using M one-dimensional Fourier Transforms. This intermediate image is then transformed into the final image, again using N one-dimensional Fourier Transforms. Expressing the two-dimensional Fourier Transform in terms of a series of $M \times N$ one-dimensional transforms decreases the number of required computations.

Even with these computational savings, the ordinary one-dimensional DFT has $M \times N$ complexity. This can be reduced to $M \log_2 N$ if the FFT is employed to compute the one-dimensional DFTs. This is a significant improvement, in particular for large images. There are various forms of the FFT and most of them restrict the size of the input image that may be transformed, often to squared images of width and height $N = n^2$ where n is an integer. The mathematical details are well described in the literature and shall not be discussed any further.

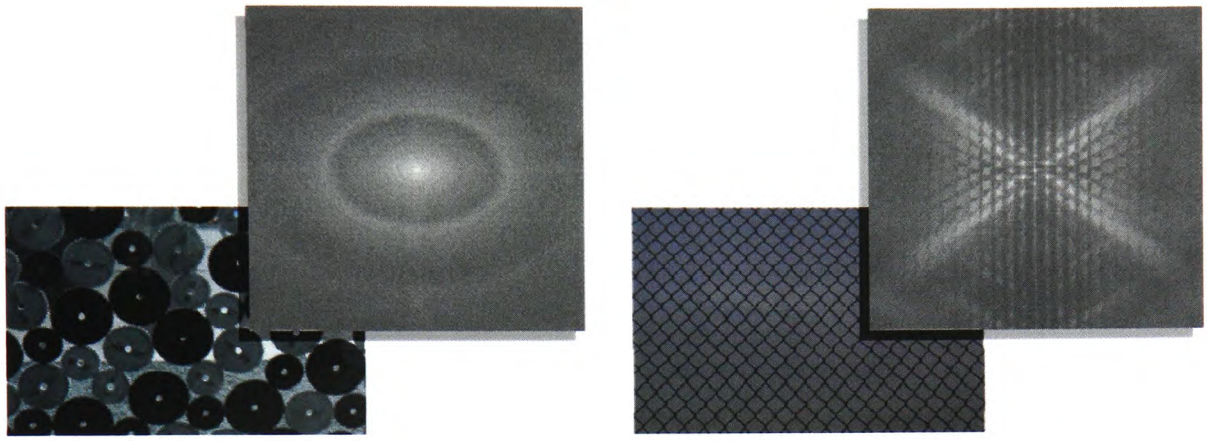
The Fourier Transform produces a complex number valued output image which can be displayed with two images, either with the real and imaginary part or with magnitude and phase. In image processing, often only the magnitude of the Fourier Transform is displayed, as it contains most of the information of the geometric structure of the spatial domain image. However, if an Fourier image shall be re-transformed into the correct spatial domain after some processing in the frequency domain, both magnitude and phase of the Fourier image must be preserved.

The Fourier Transform can be used to access the geometric characteristics of a spatial domain image. Because the image in the Fourier domain is decomposed into its sinusoidal components, it is easy to examine or process certain frequencies of the image, thus influencing the geometric structure in the spatial domain.

In the implementation for the Texture Classification System as in most other implementations the Fourier image is shifted in such a way that the DC-value (i. e. the image

mean) $F(0,0)$ is displayed in the centre of the image. The further away from the centre an image point is, the higher is its corresponding frequency.

The images in Fig. 27 show the special properties of a transformed image very well.



(a) Gearwheels (b) Wire netting

Figure 27: Fourier transformed images using the FFT.

One approach of using the Discrete Cosine Transform (DCT) for feature extraction is described in [SA04].

Related transformations are the Discrete Sine and Cosine Transforms (DST and DCT), the Walsh and the Hadamard Transforms.[6]

2.4.10 Hough Transformation

The Hough transformation can be used to isolate image features of a particular shape. The desired features have to be specified in some parametric form. Most commonly the Hough transformation is used for the detection of regular curves such as lines, circles, ellipses, etc., even though it can be employed in applications where a simple analytic description of a feature is not possible. The algorithms for complex features are computationally demanding, therefore most implementations only handle lines or circles. The advantage of the Hough transformation technique is its robustness against gaps in feature boundaries and image noise. The forward Hough transformation converts lines in a binary input image

[6]The DCT and the FHT (Fast Hadamard Transform) are available for the Texture Classification System.

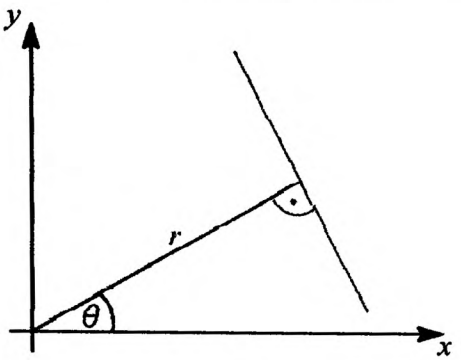


Figure 28: Sketch of a line with its r and θ values.

to points in an output image. It generates a greyscale image output where each output pixel value represents the number of pixels on the corresponding line in the input image. In the output image the x-axis represents the angle of the normal vector to the line in the input image with respect to the x-axis. The y-direction represents the radial distance of the line from the centre of the input image in image pixel units. The height of the transform image is approximately $\sqrt{W^2 + H^2}$, where W is the width and H the height of the input image. Radial distance zero is at one-half the height of the Hough transformed output image, and positive radial distances extend upward (negative y) from that point.

Analytically a line segment can be described in a number of forms, e. g. using the parametric or normal notion:

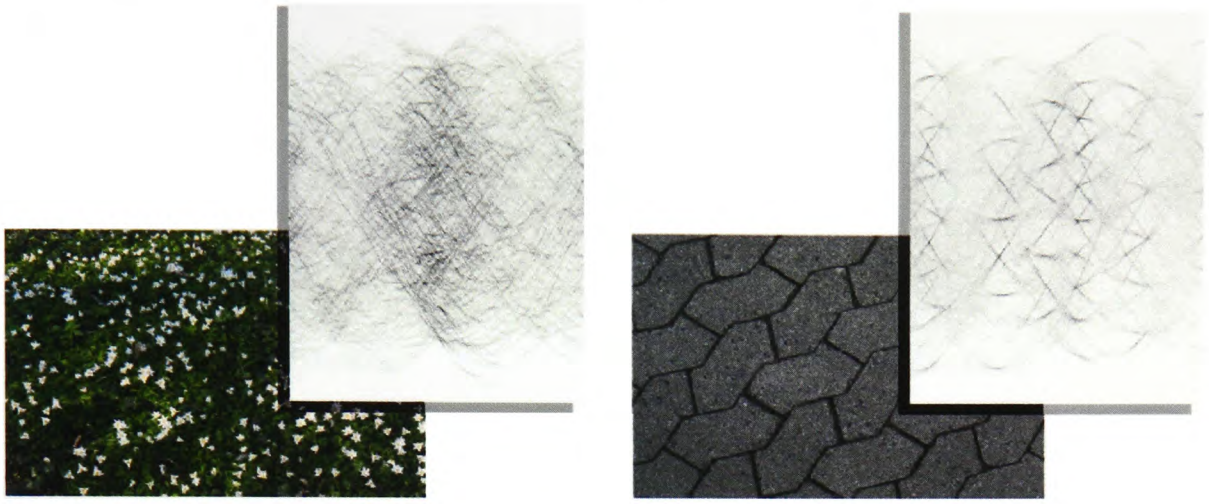
$$x \cos\theta + y \sin\theta = r \tag{5}$$

with r being the length of a normal from the origin to the line in question and θ is the angle between r and the x-axis. The values for r and θ are constant for all points (x_i, y_j) of the line. (See Fig. 28)

In images the coordinates of points (x_i, y_j) of lines, which could be edge segments, are known. They are the constants in the equation with r and θ being the unknown variables. The possible (r, θ) values defined by each (x_i, y_j) of the cartesian image space map to curves in the polar Hough parameter space. This point-to-curve transformation is the Hough transformation for straight lines. Points which are collinear in the cartesian image space are curves which intersect at a common point (r, θ) in the Hough parameter space.

For the use in image processing the Hough parameter space is quantised into finite intervals, so called accumulator cells. Each (x_i, y_j) is transformed into a discretised (r, θ)

curve. The accumulator cells along this curve are incremented and the resulting peaks in the accumulator array represent a corresponding straight line.



(a) Small flowers

(b) Interlocking pavement stones

Figure 29: Images transformed with the forward Hough transformation.

The same procedure can be used to detect other features. For the case of circles the parametric equation is:

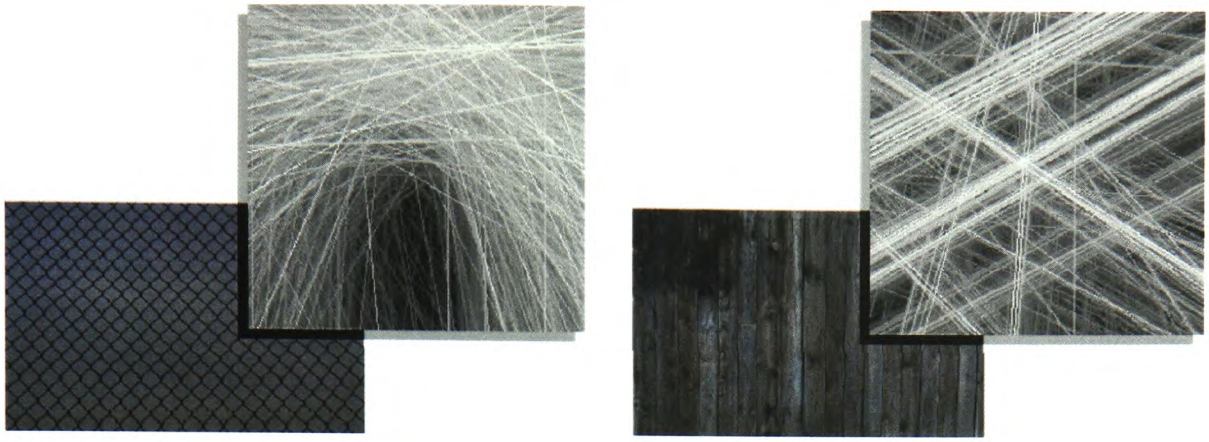
$$(x-a)^2 + (y-b)^2 = r^2 \tag{6}$$

with (a, b) being the centre of the circle, r its radius.

The images in Fig. 29 show the results using lines as features. The differences in the transformed images are fairly obvious and can be utilised for further exploration.

The reverse Hough transformation converts points in the input image to lines in the output image. It is not a true inverse in that when it is applied to a Hough transformed image it does not reproduce the original image. The relationship of pixels in the transformed image to lines in the reverse transformed image is the inverse of that described for the forward transformation.

For the Texture Classification System the reverse Hough transformation has been employed to transform the original image. This is certainly most unusual and not the intended usage of the algorithm, but the results show a certain distinctness between different textures. Therefore it makes sense to extract those features. Fig. 30 shows two examples.



(a) Wire netting (b) Wooden fence

Figure 30: Result of the reverse Hough transformation applied to an image.

2.4.11 Wavelet Transformation

2.4.11.1 The Short-Time Fourier Transformation

To understand the Wavelet Transformation (WT) it is best to look at the Short-Time Fourier Transformation (STFT) first. The idea behind the STFT is that even a non-stationary signal is stationary for a short period. If now every segment is Fourier transformed separately a localisation is being achieved.[7] Therefore a window function is added to the Fourier Transformation (FT). The window size must be equal to the size of the stationary signal. The complete signal, or image, is being looked at through a moving window.

The equation for the STFT is:

$$STFT_x^w(\tau, f) = \int_t x(t)w^*(t-\tau)e^{-j2\pi ft}dt \tag{7}$$

with $w(t)$ being the window-function, and $w^*(t)$ is the complex conjugated window-function. For each value τ and f new coefficients are being calculated, with τ being the shifting parameter and f the frequency.

The window function itself can be a simple square or for example a gaussian function. In the case of the gaussian window function the STFT is also called Gabor function.

[7]Usually one-dimensional signals over time are being addressed, hence the name.

The drawback of the STFT is its problem with the Heisenberg's uncertainty relation. It is not possible to tell the exact frequency of a signal at an exact point of time. This means that with smaller windows and better time resolution the frequency resolution decreases.

2.4.11.2 The Continuous Wavelet Transformation

The WT was introduced to overcome the problems of the STFT. As there is no way past Heisenberg the 'trick' is to use windows of different size for a transformation. A small window is being used for high frequencies to localise the time, or position in the case of an image, and a wide window for lower frequencies, that usually are distributed over the whole signal and are important for the overall appearance.

The Continuous Wavelet Transformation (CWT) is being described by the following equation:

$$CWT_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int_t x(t)\psi^* \frac{t-\tau}{s} dt \tag{8}$$

with τ being the shifting parameter and s the scaling parameter. The factor $\frac{1}{\sqrt{|s|}}$ is used to normalise the energy for the different scales.

$\psi(t)$ is the transformed function, the so called 'mother wavelet'. All other window functions are being derived from this mother wavelet.

There is no frequency in the Eqn. 8. This place has been taken by the scaling factor s, which for simplicity can be seen as being inverse proportional to the frequency.

The main difference to the STFT is that for the WT the window size and the frequency are both dependent on the same scaling factor s.

More information on the CWT can be found in [Gra95a, Kai94, Kle96, LMR94].

2.4.11.3 Wavelet Types

The wavelets used for transformation can be classed into families.

The first ever wavelet is the Haar wavelet[8], discovered by A. HAAR in 1909. In the 1930s it was used by PAUL LEVY to analyze the Brown movement of atoms. At that time it was called 'Haar-basis-function'.

[8]The word 'wavelet', standing for 'small wave', was not used at that time but introduced by MORLET in the 1980s.

The Haar wavelet belongs to the Daubechies wavelet family, introduced by INGRID DAUBECHIES in the 1980s. The members of that family differ in the number of filter coefficients. The Haar wavelet is identical to the Daubechies wavelet with 2 coefficients and therefore also called Daubechies-2-wavelet.

Next to the Daubechies wavelet functions there are a great number of different wavelet functions existing, some of which are the B-spline- (= Chui-Wang semi-orthogonal spline wavelets), Coiflet-, Littlewood-Paley-Stein-, Meyer-, Morlet-, Remez-, Spline-, Symlet- and Tryme-Wavelets to name some of the better known types. The B-spline-, Coiflet- and Daubechies-Wavelets have already been implemented for the Texture Classification System.

Wavelets also differ by being orthogonal or bi-orthogonal. The orthogonal wavelets are recursively calculable and lead to fast, discrete algorithms. Also the forward and reverse transformation can use the same wavelet. The drawback is that the orthogonal wavelets lead to a set of non-linear equations, which usually can only be solved numerically. The bi-orthogonal wavelets are symmetric and need different wavelets for forward and reverse transformation.

Fig. 31 shows some of the mother wavelets.

2.4.11.4 The Discrete Wavelet Transformation

For the use in computers the discretised form of the WT is needed. Discretising is usually done by regular sampling. In the case of the Discrete Wavelet Transformation (DWT) it is possible to reduce the sampling rate by changing the scaling factor s. According to NYQUIST the result for a high value of s is a low frequency resolution. (See also the Glossary or [vdEV90])

According to SHANNON [Sha48] the sampling frequency can be reduced if only accordingly reduced spatial frequencies shall be detected. The sampling rate N can therefore be calculated by:

$$N_{i+1} = \frac{s_i}{s_{i+1}} N_i \tag{9}$$

with s being the scaling factor and $s_{i+1} > s_i$, and $N_{i+1} < N_i$. The reduced sampling rate saves computing resources.

For the practical realisation a logarithmic scaling is being used, in most cases the base two logarithm. This is helpful for the implementation using digital filters.

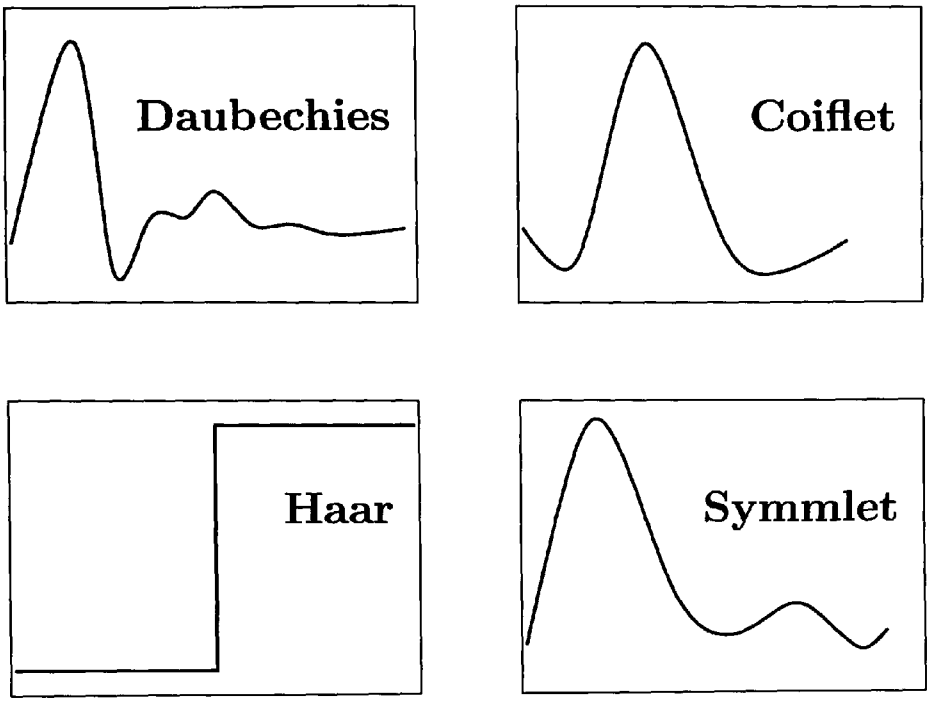


Figure 31: Sketches of mother wavelets.

2.4.11.5 The Multi-Scale Analysis

The practical realisation of the DWT analyzes in contrast to the CWT not single frequencies, but frequency bands. These bands stretch between the logarithmic scale values.

The algorithm that calculates the DWT is called Pyramidal Coding Algorithm (PCA) or Multi-Scale Algorithm (MSA). The wavelet itself can be regarded as a band-pass filter which analyzes a certain frequency band. As the frequencies are being halved[9] from one step to the next, high-pass and low-pass filters can be used.

For a sampling frequency of 2π and the according highest signal frequency of π the cut-off frequency of the low-pass filter and the threshold frequency of the high-pass filter is $\frac{\pi}{2}$. The filters are being derived from the mother wavelet.

The algorithm runs as follows: the input signal, which holds frequencies from 0 to π, is being halved by high-pass ($\frac{\pi}{2}$ to π) and low-pass (0 to $\frac{\pi}{2}$) filters. The output of the high-pass filter are called 'level 1 DWT coefficients'. The low-pass frequency band is then halved again, producing the 'level 2 DWT coefficients' for the then high-pass frequency band ($\frac{\pi}{4}$ to $\frac{\pi}{2}$). This can be repeated for further levels.

[9]Due to the logarithmic scaling the scaling factor doubles at every step.

As in every level half the information is extracted the sampling rate can be halved, too. This is called downsampling and reduces the amount of data without loosing information. Fig. 32 shows the principle.

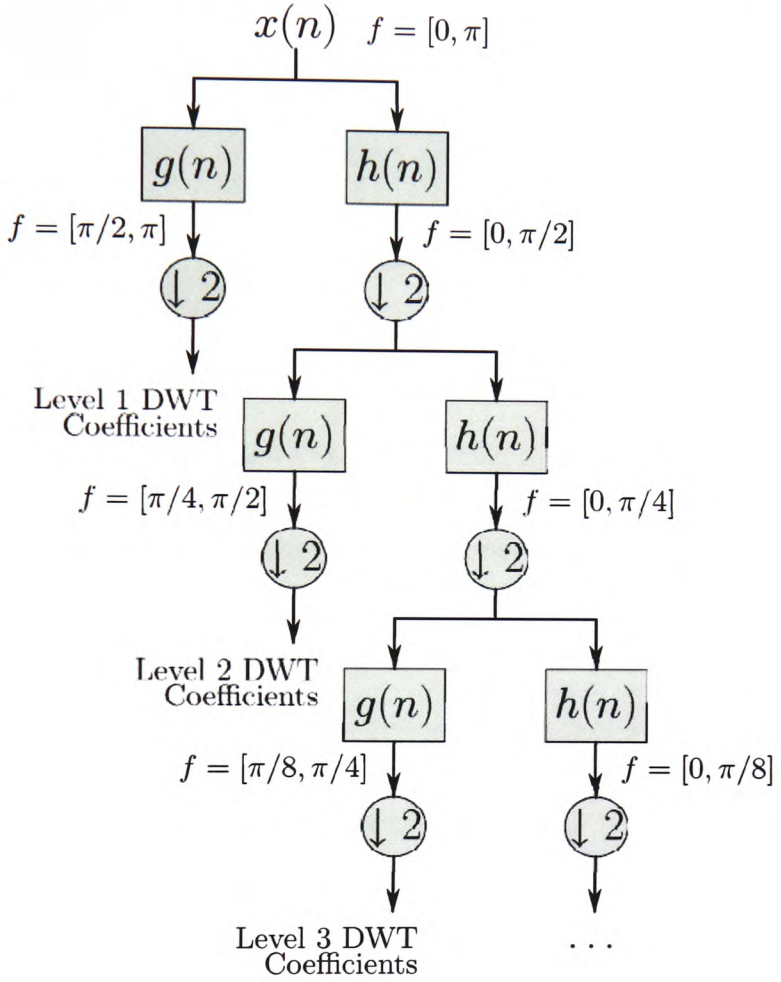


Figure 32: Multi-scale analysis. $x(n)$ is the input signal with frequencies from 0 to π. $g(n)$ is the high-pass, $h(n)$ the low-pass filter. $\downarrow 2$ shows the process of downsampling.

In the case of two-dimensional signals, i. e. images, the DWT is calculated over the rows and columns separately. This leads to four subimages for every level with three high-pass filtered subimages, named the 'horizontal-detailed-subband', 'vertical-detailed-subband', and 'diagonal-detailed-subband'. The fourth subimage is the 'low-pass-residue' which is the input for the next iteration.

Further information on WT can be found in [BB97, BS95, Don92, Gra95b, LF93, LSdWD97, NS95, Rei96, STG96] and in [Mül90a, Mül90b, SMM94] for specialised FT.

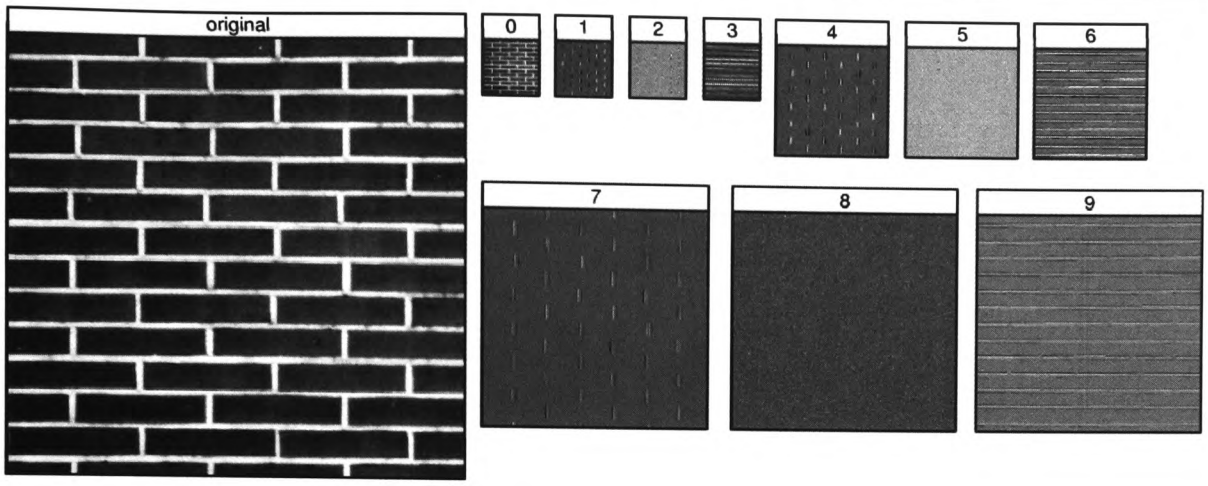


Figure 33: 3rd level WT with 2nd order Daubechies filter

Figure 33 shows a DWT of level three for which a second order Daubechies filter was used. The subimage '0' shows the low-pass residue and subimages '1–9' the high-pass residues of the first, second and third level respectively.

2.4.12 LAWS Micro Structures

A further approach is using the LAWS metrics for texture feature extraction [Law80a, Law80b]. Features are extracted from the image by convoluting the original image with a spatial filter. The spatial filters are comprised of 5×5 kernels[10] derived from centre-weighted vectors defined by the LAWS texture measure. By these methods microstatistical estimations are being obtained.

LAWS developed five labelled vectors which are combined to form matrices. When convoluted with a textured image these matrices extract individual structural components of the image. The five vectors are:

$$\begin{aligned} \text{L5} &= [\ 1,\ 4,\ 6,\ 4,\ 1] \\ \text{E5} &= [-1,-2,\ 0,\ 2,\ 1] \\ \text{S5} &= [-1,\ 0,\ 2,\ 0,-1] \\ \text{W5} &= [-1,\ 2,\ 0,-2,\ 1] \\ \text{R5} &= [\ 1,-4,\ 6,-4,\ 1] \end{aligned}$$

[10]Early versions used 3×3 kernels. Ojala et al. describe the use of those for texture classification in [OPN96].

The letters are mnemonics for Level, Edge, Spot, Wave, and Ripple. Note that all kernels except L5 are zero-sum.

With these vectors the 25 matrices shown in Tab. 5 are created. The matrices are then used as convolution kernels to generate 25 new images from one texture image.

×	L5	E5	S5	W5	R5
L5	1 4 6 4 1 4 16 24 16 4 6 24 36 24 6 4 16 24 16 4 1 4 6 4 1	-1 -4 -6 -4 -1 -2 -8 -12 -8 -2 0 0 0 0 0 2 8 12 8 2 1 4 6 4 1	-1 -4 -6 -4 -1 0 0 0 0 0 2 8 12 8 2 0 0 0 0 0 -1 -4 -6 -4 -1	-1 -4 -6 -4 -1 2 8 12 8 2 0 0 0 0 0 -2 -8 -12 -8 -2 1 4 6 4 1	1 4 6 4 1 -4 -16 -24 -16 -4 6 24 36 24 6 -4 -16 -24 -16 -4 1 4 6 4 1
E5	-1 -2 0 2 1 -4 -8 0 8 4 -6 -12 0 12 6 -4 -8 0 8 4 -1 -4 0 2 1	1 2 0 -2 -1 2 4 0 -4 -1 0 0 0 0 0 -2 -4 0 4 2 -1 -2 0 2 1	1 2 0 -2 -1 0 0 0 0 0 -2 -4 0 4 2 0 0 0 0 0 1 2 0 -2 -1	1 2 0 -2 -1 -2 -4 0 4 2 0 0 0 0 0 2 4 0 -4 -2 -1 -2 0 2 1	-1 -2 0 2 1 4 8 0 -8 -4 -6 -12 0 12 6 4 8 0 -8 -4 -1 -4 0 2 1
S5	-1 0 2 0 -1 -4 0 8 0 -4 -6 0 12 0 -6 -4 0 8 0 -4 -1 0 2 0 -1	1 0 -2 0 1 2 0 -4 0 2 0 0 0 0 0 -2 0 4 0 -2 -1 0 2 0 -1	1 0 -2 0 1 0 0 0 0 0 -2 0 4 0 -2 0 0 0 0 0 1 0 -2 0 1	1 0 -2 0 1 -2 0 4 0 -2 0 0 0 0 0 2 0 -4 0 2 -1 0 2 0 -1	-1 0 2 0 -1 4 0 -8 0 4 -6 0 12 0 -6 4 0 -8 0 4 -1 0 2 0 -1
W5	-1 2 0 -2 1 -4 8 0 -8 4 -6 12 0 -12 6 -4 8 0 -8 4 -1 2 0 -2 1	1 -2 0 2 -1 2 -4 0 4 -2 0 0 0 0 0 -2 4 0 -4 2 -1 2 0 -2 1	1 -2 0 2 -1 0 0 0 0 0 -2 4 0 -4 2 0 0 0 0 0 1 -2 0 2 -1	1 -2 0 2 -1 -2 4 0 -4 2 0 0 0 0 0 2 -4 0 4 -2 -1 2 0 -2 1	-1 2 0 -2 1 4 -8 0 8 -4 -6 12 0 -12 6 4 -8 0 8 -4 -1 2 0 -2 1
R5	1 -4 6 -4 1 4 -16 24 -16 4 6 -24 36 -24 6 4 -16 24 -16 4 1 -4 6 -4 1	-1 4 -6 4 -1 -2 8 -12 8 -2 0 0 0 0 0 2 -8 12 -8 2 1 -4 6 -4 1	-1 4 -6 4 -1 0 0 0 0 0 2 -8 12 -8 2 0 0 0 0 0 -1 4 -6 4 -1	-1 4 -6 4 -1 2 -8 12 -8 2 0 0 0 0 0 -2 8 -12 8 -2 1 -4 6 -4 1	1 -4 6 -4 1 -4 16 -24 16 -4 6 -24 36 -24 6 -4 16 -24 16 -4 1 -4 6 -4 1

Table 5: The kernels for the LAWS 5 × 5 texture measures.

The images[11] in Fig. 34 to 37 show very well, that depending on the type of texture special texture features are enhanced by certain LAWS filters and therefore can be utilised for further processing.

2.4.12.1 LAWS Texture Energy Measures

While it is possible to use the 25 (or only some of them) new LAWS filtered images for further processing by applying some of the above mentioned filters or transformations, or

[11] For better printing results all but the L5L5 subimage are displayed negative and equalised.

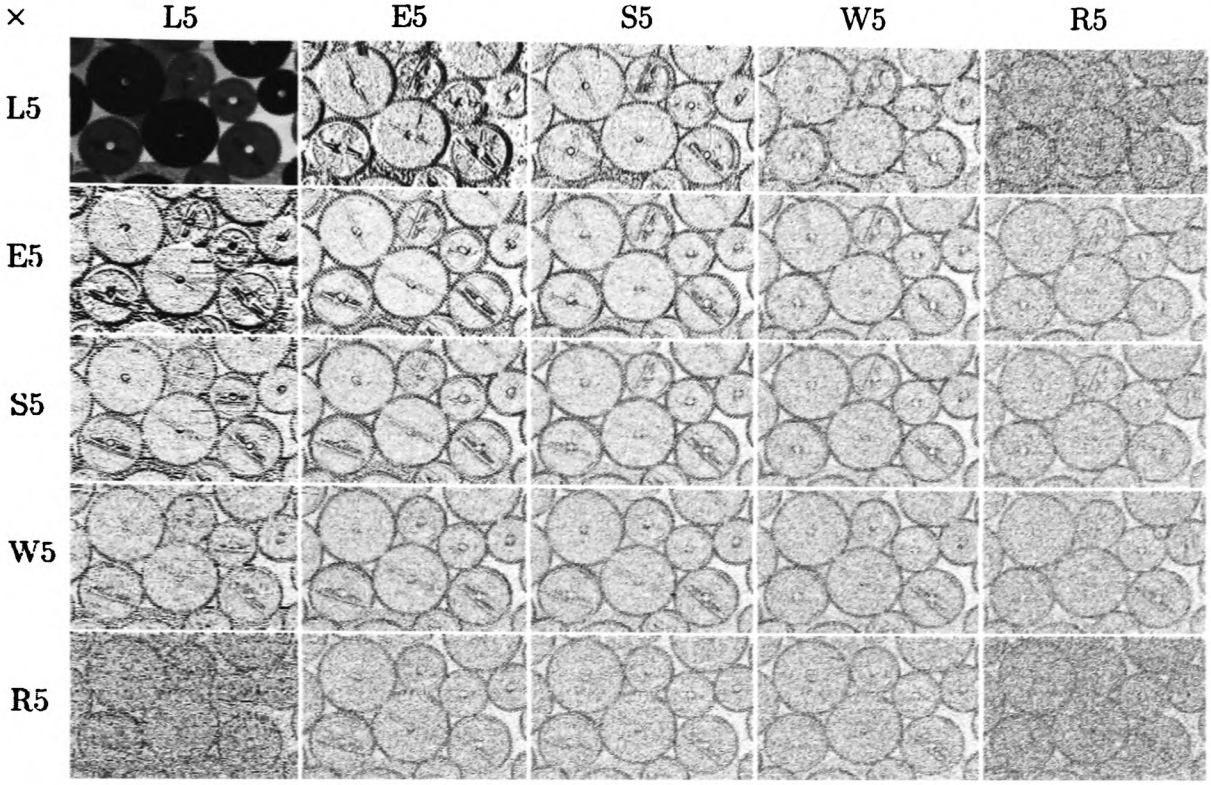


Figure 34: The LAWS texture measures of image 114.20, gearwheels.

by extracting statistical data as described in Section 2.5, it is also an option to calculate the Texture Energy Measures (TEM) as described by LAWS [Law80a, Law80b].

These measures are computed by first applying the convolution kernels to the texture image, and then performing a nonlinear windowing operation. The windowing operation replaces every pixel $P_{i,j}$ in the 25 LAWS filtered images with a texture energy measure $T_{i,j}$. This measure is obtained by the following neighbourhood function:[12]

$$T_{i,j} = \sum_{i=-7}^{7} \sum_{j=-7}^{7} |P_{i,j}| \tag{10}$$

LAWS also suggests the use of another filter instead of the 'absolute value windowing' filter of Eqn. 10:

$$T_{i,j} = \sqrt{\sum_{i=-7}^{7} \sum_{j=-7}^{7} {P_{i,j}}^2} \tag{11}$$

At this point 25 TEM images are generated from the original texture image.

[12]The window size 15 × 15 is not mandatory but just an example.

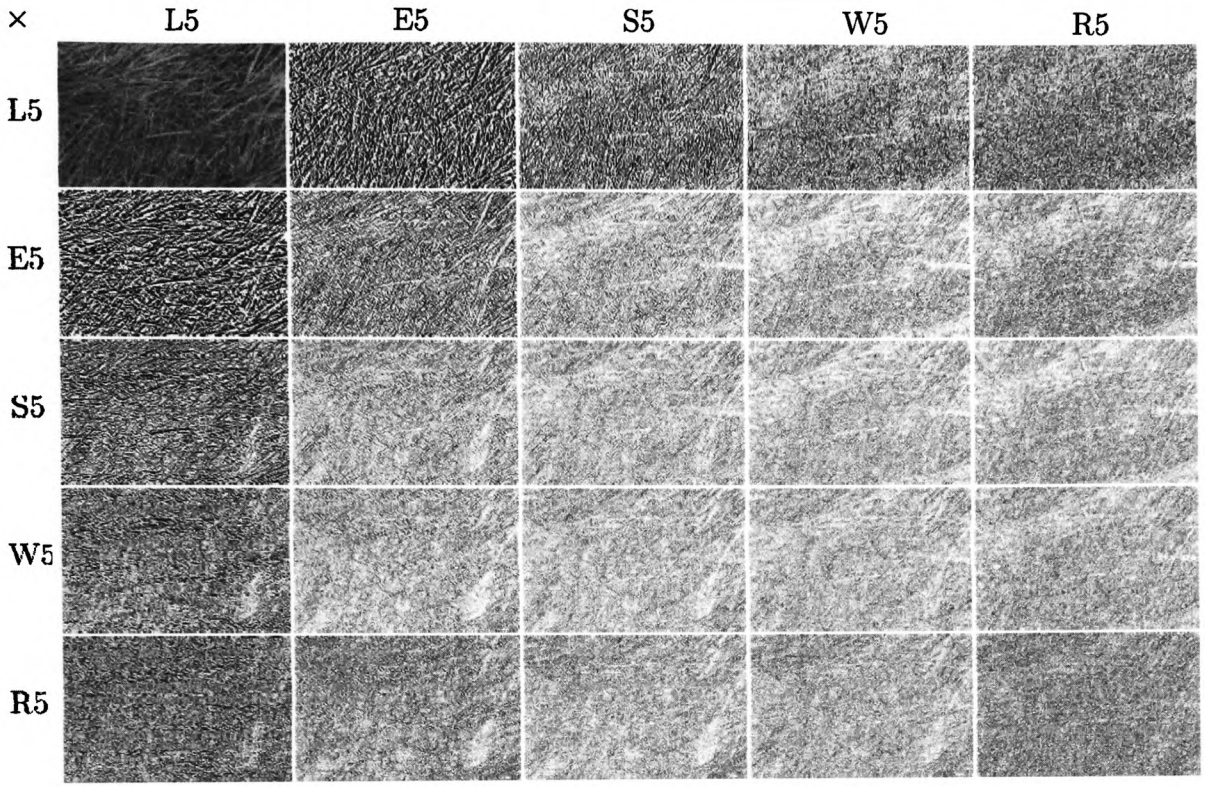


Figure 35: The LAWS texture measures of image 114.22, dried grassblades.

All convolution kernels used so far are zero-mean with the exception of the L5L5 kernel. In accordance with Laws' suggestions, this can therefore be use as a normalisation image. Normalising any TEM image pixel-by-pixel with the L5L5 TEM image will normalise that feature for contrast.

If the direction of a texture is not desired, similar features can be combined. For example the E5L5 kernel is sensitive to horizontal, the L5E5 to vertical edges. Combining these two features by adding the TEM images gives an edge sensitive result, which is rotational invariant.

This is possible for the following TEM images:

$$\mathrm{E5L5}_{\mathrm{rot}}^{\mathrm{TEM}} = \mathrm{E5L5}^{\mathrm{TEM}} + \mathrm{L5E5}^{\mathrm{TEM}} \quad (12)$$

$$\mathrm{S5L5}_{\mathrm{rot}}^{\mathrm{TEM}} = \mathrm{S5L5}^{\mathrm{TEM}} + \mathrm{L5S5}^{\mathrm{TEM}} \quad (13)$$

$$\mathrm{W5L5}_{\mathrm{rot}}^{\mathrm{TEM}} = \mathrm{W5L5}^{\mathrm{TEM}} + \mathrm{L5W5}^{\mathrm{TEM}} \quad (14)$$

$$\mathrm{R5L5}_{\mathrm{rot}}^{\mathrm{TEM}} = \mathrm{R5L5}^{\mathrm{TEM}} + \mathrm{L5R5}^{\mathrm{TEM}} \quad (15)$$

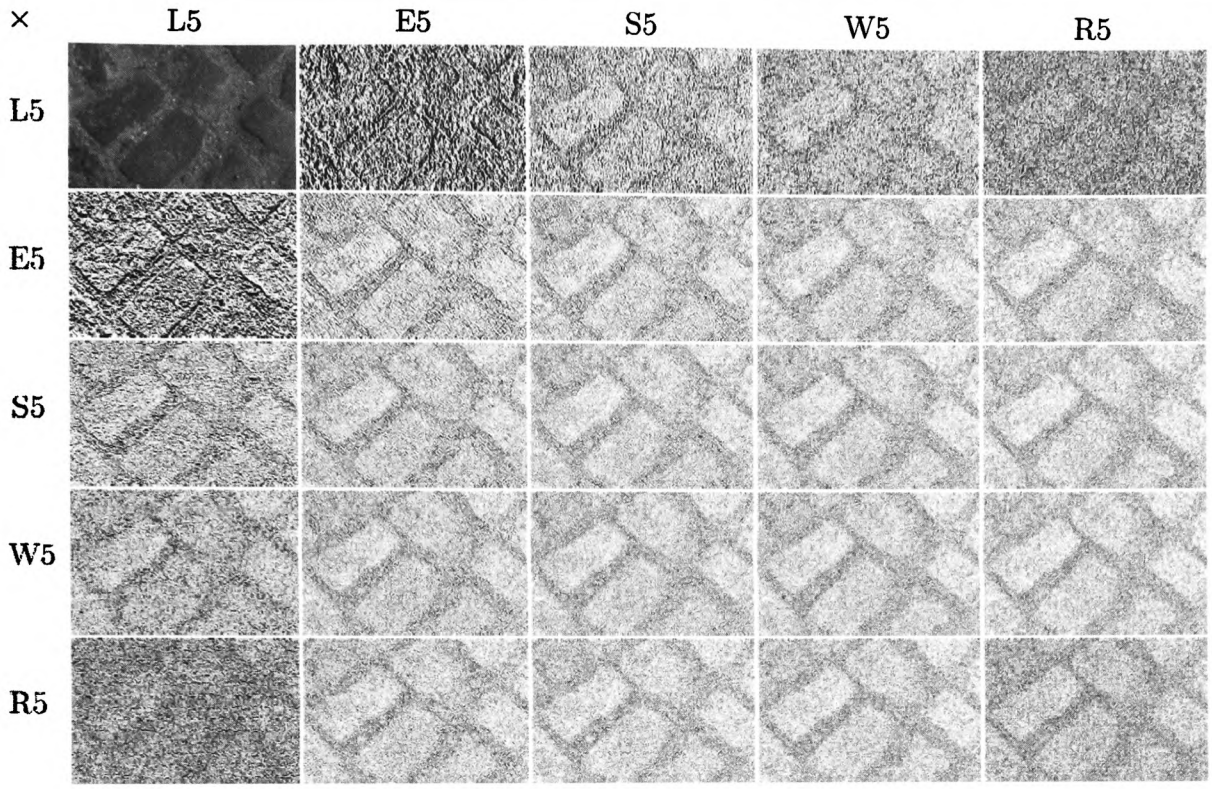


Figure 36: The LAWS texture measures of image 114.34, old pavement.

$$S5E5^{TEM}_{rot} = S5E5^{TEM} + E5S5^{TEM} \quad (16)$$

$$W5E5^{TEM}_{rot} = W5E5^{TEM} + E5W5^{TEM} \quad (17)$$

$$R5E5^{TEM}_{rot} = R5E5^{TEM} + E5R5^{TEM} \quad (18)$$

$$W5S5^{TEM}_{rot} = W5S5^{TEM} + S5W5^{TEM} \quad (19)$$

$$R5S5^{TEM}_{rot} = R5S5^{TEM} + S5R5^{TEM} \quad (20)$$

$$R5W5^{TEM}_{rot} = R5W5^{TEM} + W5R5^{TEM} \quad (21)$$

The remaining features are scaled by 2 to keep all features consistent with respect to size:

$$L5L5^{TEM}_{rot} = L5L5^{TEM} * 2 \quad (22)$$

$$E5E5^{TEM}_{rot} = E5E5^{TEM} * 2 \quad (23)$$

$$S5S5^{TEM}_{rot} = S5S5^{TEM} * 2 \quad (24)$$

$$W5W5^{TEM}_{rot} = W5W5^{TEM} * 2 \quad (25)$$

$$R5R5^{TEM}_{rot} = R5R5^{TEM} * 2 \quad (26)$$

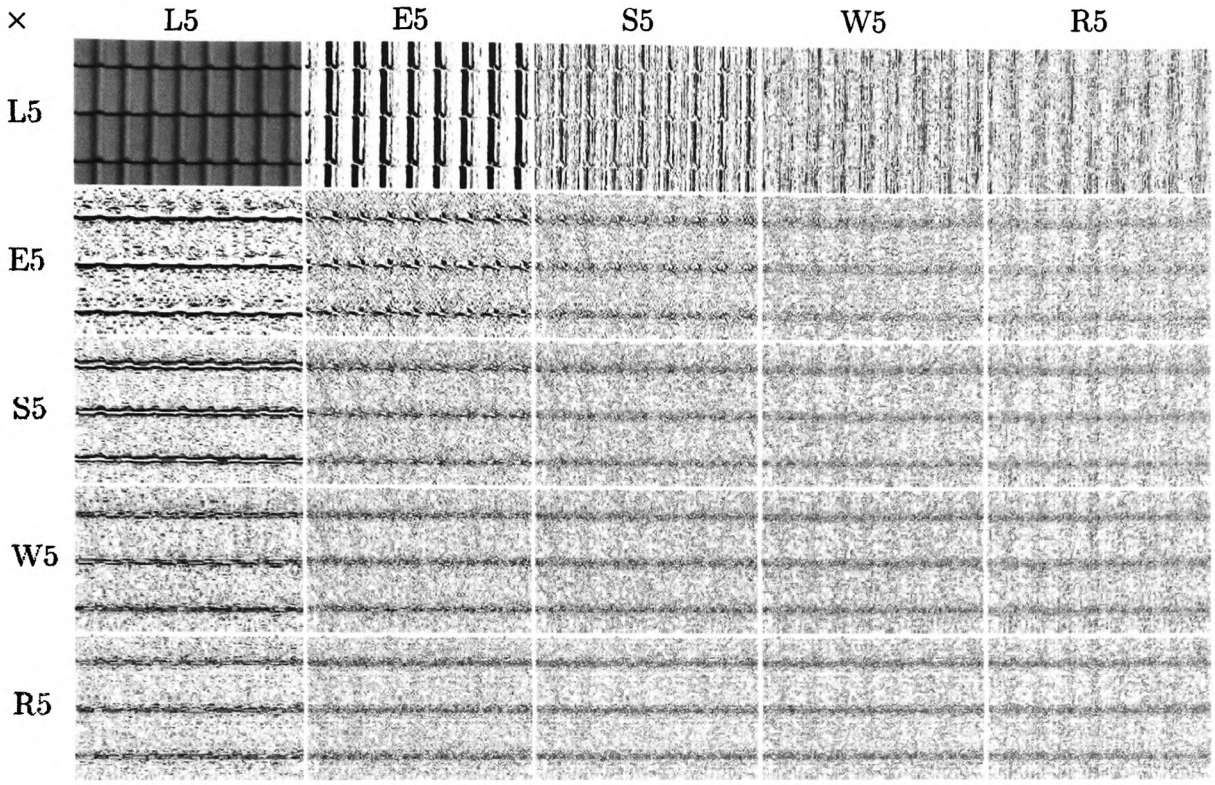


Figure 37: The LAWS texture measures of image 114.40, new rooftiles.

The result is a set of 15 texture features which are rotational invariant. Combining these images gives a 15-element feature vector for every pixel of the original texture image.

In the Texture Classification System usually texture features of subimages are being calculated. As these subimages comprise of 100 to 1.000 pixel using all TEM features of all pixels is not advisable. Better results can be obtained by applying the statistical operators described in Section 2.5 first.

2.4.13 Other Filters and Transformations

To end the chapter about filters and transformations it has to be mentioned that this is of course only an overview over possible methods that can be used for the Texture Classification System. Applying the filters and transformations described in the sections above with only some of the possible parameters set and using only few combinations will already give a great amount of specialised data for a texture image. After this data has been evaluated with statistical methods the resulting feature vector will exceed the size of

the texture image file by some magnitudes. How to reduce the size of the feature vector in a way, that only the redundant information is being discarded, will be explained in Section 2.8.

Nevertheless some other interesting methods shall be mentioned, but not described in detail.

2.4.13.1 Morphological Methods

The mathematical morphology, or short morphology, can be defined as a theory for analysing spatial structures. The word morphology yields from the aim to analyze the form of objects. It is not only a theory but a technique for image analysis.

Some of the more important methods are 'opening', 'closing', 'skeletonisation', 'watershed', 'top hat', and 'outline'. These operators are available for the Texture Classification System and some more information can be found in the Glossary on Pages 185, 190, 194, 199, 201, 203, and 209, or in [Soi98, Dus99].

2.4.13.2 Fuzzy Image Processing

As there is a lot of uncertainty in image processing, especially when addressing texture analysis, using fuzzy logic is a good option. For the author's Texture Classification System only Fuzzy Clustering has been used so far. But as TIZHOOSH describes in his book [Tiz98] fuzzy methods can be applied to filtering and segmentation. Some of the interesting methods are 'fuzzy histograms', 'fuzzy morphology', 'fuzzy noise reduction', 'fuzzy segmentation', 'fuzzy skeletonisation', and 'fuzzy edge detection'. See also [Dav89]. To the author's knowledge none of these methods have been applied for texture classification. As these methods hold great potential, it is only a question of time before they are being implemented for texture classification by fellow researchers or the author himself.

2.5 Statistical Methods

As with the filtering and transformation techniques a great number of statistical methods exist. Interestingly some of these are seldom used in the field of image processing, even though they are highly interesting and fairly easily computable. Among these, belonging to the first order statistics, are the moments of higher order, i. e. the skewness e and the kurtosis c.

2.5.1 1st Order Statistics

To calculate the histogram H of an image the number of pixels with a specific greylevel value are counted and entered in an array of size n, with n being the number of different greylevels. Statistical values can be obtained from the histogram.

From the absolute histogram distribution the relative histogram distribution or probability distribution is derived:

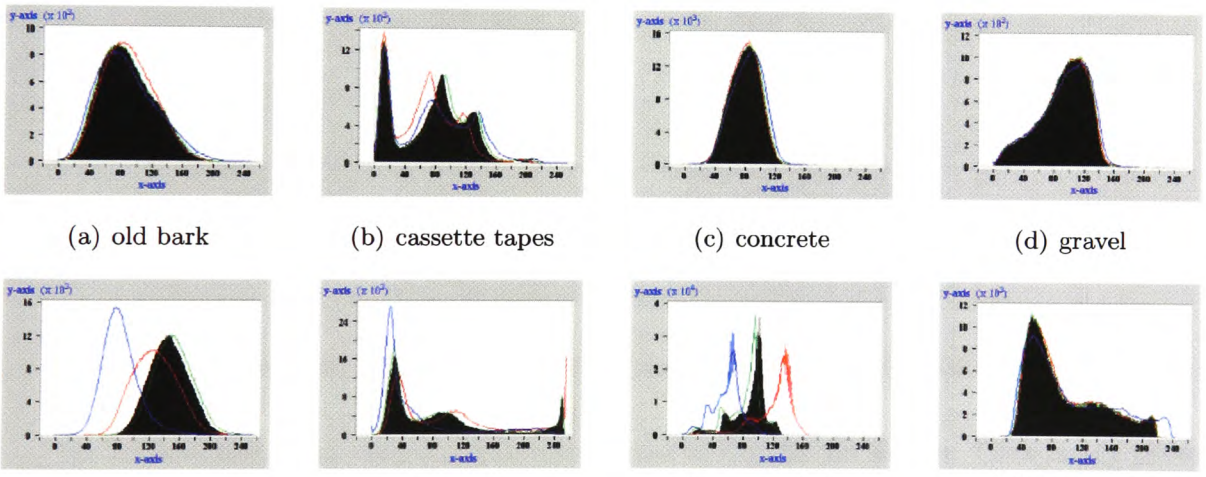
$$h(z) = \frac{H(z)}{N} \tag{27}$$

and

$$\sum_{z=z_{min}}^{z_{max}} h(z) = 1 \tag{28}$$

N is the number of image points, z_{min} the smallest, z_{max} the largest greylevel value.

It must be mentioned that the statistical features obtained from the histograms hold no information concerning the contents of an image. They are features describing the greylevel distribution of the image.



(a) old bark (b) cassette tapes (c) concrete (d) gravel

(e) moss (f) reeds (g) new rooftiles (h) bare trees

Figure 38: Histograms of texture images (Figs. 114.5, 114.11, 114.13, 114.23, 114.28, 114.37, 114.40, 114.58) with black curve for grey level image, red, green, and blue curves for the red, green, and blue colour planes, respectively.

When the greylevel values have a tendency to cluster around a particular value, a characterisation can than be obtained by the sums of the integer powers of the values, the moments.

2.5.1.1 Mean

The 1st statistical moment, the average or mean, is obtained by:

$$M_1 = \bar{z} = \sum_{z=z_{min}}^{z_{max}} z h(z) \tag{29}$$

This is a measure for the average greylevel within an image. By using this feature darker images can be separated from lighter ones. It therefore must be ensured that the images are not equalised individually before being processed by the Texture Classification System.

As most images for texture analysis are at some stage equalised (camera, scanner, etc.) the information obtained will only be of little use.

Useful for the complete system will be the relative average of subimages within a multitexture image ("subimage A is lighter than subimage B"). Linguistic variables are needed to compute this information.

	Mean
old bark	88·345
cassette tapes	74·4758
concrete	79·9455
gravel	91·1189
moss	146·346
reeds	84·7633
new rooftiles	89·0893
bare trees	95·2389

Table 6: The mean values of the histograms in Fig. 38

In Table 6 the mean values of the histograms in Fig. 38 are shown. Comparing these values with the histogram images one can see, that the mean does not give the position of the most often occurring greylevel, i. e. h_{max}, but the centre of gravity for the distribution.

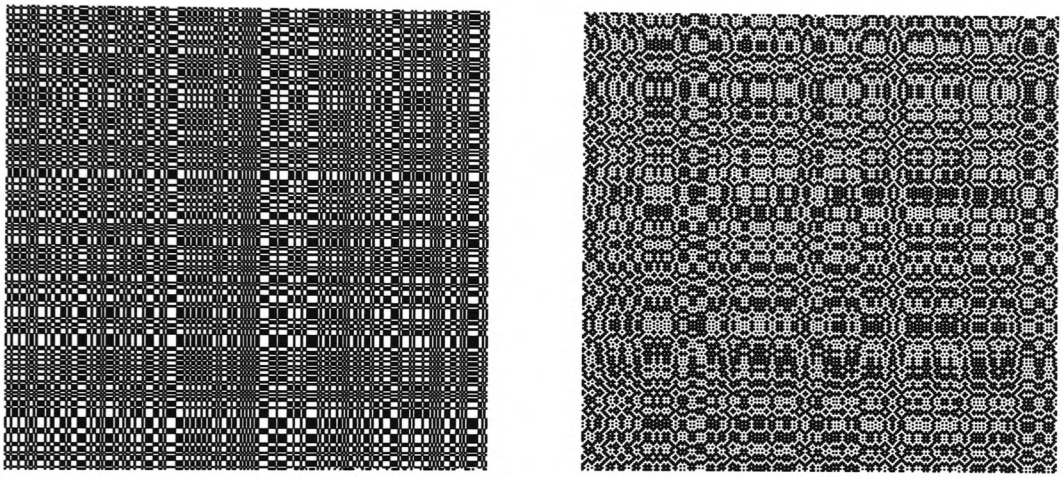
2.5.1.2 Variance

The 2nd statistical moment, the variance:

$$M_2 = \sigma^2 = \sum_{z=z_{min}}^{z_{max}} (z - \bar{z})^2 h(z) \tag{30}$$

Variance	
old bark	1196·73
cassette tapes	2135·71
concrete	380·311
gravel	1059·29
moss	682·219
reeds	4516·21
new rooftiles	535·856
bare trees	2407·37

Table 7: The variance values of the histograms in Fig. 38



(a) (b)

Figure 39: Images **even** and **odd**

The variance is the mean squared-deviation of the values from their mean value. It characterises the width or variability around that value.

In Table 7 the variance values of the histograms in Fig. 38 are shown. The histogram images show that a wide distribution of greylevels give a high variance. Especially values further away from the mean value are emphasised. Textures which show differences in their distribution of greylevels can be distinguished by the variance feature value.

Even though the variance is quite a useful measure it is not the 'non-plus-ultra' feature for which it is sometimes being regarded. The images in figure 39 have identical mean and variance values.

For this reason it is necessary to obtain more features. The histogram holds more features, e. g. the skewness and the kurtosis.

2.5.1.3 Skewness

The 3rd statistical moment, the skewness:

$$M_3 = e = \sum_{z=z_{min}}^{z_{max}} (z - \bar{z})^3 h(z) \tag{31}$$

The skewness is a measure for the asymmetry of a distribution. If the third moment in dimensionless form

$$M_3' = \frac{M_3}{\sqrt{{M_2}^3}} \tag{32}$$

is positive, the distribution is tilted to the left in comparison to the gaussian form, otherwise to the right.

Skewness	
old bark	0·354898
cassette tapes	0·0535386
concrete	−0·103594
gravel	−0·603006
moss	0·212309
reeds	1·19932
new rooftiles	−1·09874
bare trees	0·838159

Table 8: The skewness values of the histograms in Fig. 38

In Table 8 the skewness values of the histograms in Fig. 38 are shown. Good examples are the histograms 38(a) and 38(d). Skewness values are a feature to distinguish between textures with different asymmetric pixel brightness distributions.

2.5.1.4 Kurtosis

The 4th statistical moment, the kurtosis:

$$M_4 = c = \sum_{z=z_{min}}^{z_{max}} (z - \bar{z})^4 h(z) \tag{33}$$

The kurtosis is a measure for the deviation of the maximum towards larger or smaller values, i.e. a peakedness or flatness of the distribution. If the fourth moment in dimensionless form

$$M_4' = \frac{M_4}{{M_2}^2} \tag{34}$$

is positive, the maximum of the distribution is larger in comparison to the gaussian form, otherwise it is smaller. A positive kurtosis is also termed leptokurtic, a negative platykurtic. If the kurtosis value is near to zero the distribution is called mesokurtic.

Kurtosis	
old bark	−0·385198
cassette tapes	−0·803424
concrete	−0·48555
gravel	−0·32211
moss	0·0475487
reeds	0·473981
new rooftiles	1·04514
bare trees	−0·407307

Table 9: The kurtosis values of the histograms in Fig. 38

In Table 9 the kurtosis values of the histograms in Fig. 38 are shown. Good examples are the histograms 38(b) (platykurtic), 38(e) (mesokurtic) and 38(g) (leptokurtic). Kurtosis values are a feature to distinguish between textures with different maximum levels compared to the gaussian distribution.

2.5.1.5 Entropy

The entropy is according to [Hab91] a measure for the mean information contents of an image, i. e. the amount of information per pixel can be calculated from the relative number $h(z)$ of pixels of that greylevel.

The entropy is being calculated from the greylevel histogram by

$$E = -\sum_{z=z_{min}}^{z_{max}} h(z) \log_2 h(z) \tag{35}$$

with $h(z)$ being the probability of a pixel having the greylevel value z.

In Table 10 the entropy values of the of the histograms in Fig. 38 are shown.

Entropy	
old bark	−4·93604
cassette tapes	−4·97902
concrete	−4·37892
gravel	−4·8198
moss	−4·67674
reeds	−5·06672
new rooftiles	−4·31558
bare trees	−5·04311

Table 10: The entropy values of the histograms in Fig. 38

2.5.2 2nd Order Statistics

2.5.2.1 Cooccurrence Matrix

The Cooccurrence Matrix (CM) provides a large amount of features for the feature vector and gives information about the orientation of the texture. Among these features the energy E_d, contrast K_d and entropy H_d are some of the more important ones.

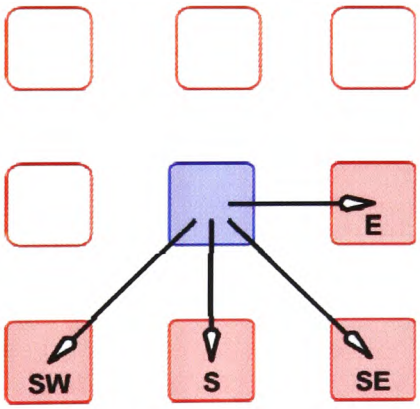


Figure 40: The cooccurrence matrix principle.

Fig. 40 describes the principle idea behind the CM. Neighbouring pixels are being tested for their greylevel value. The two greylevel values make an (x, y) pair and the according (x, y) point in the CM is incremented by one. The height and width of the CM equals the maximal numbers of greylevels of the texture image, usually 256×256 for 8-bit images.

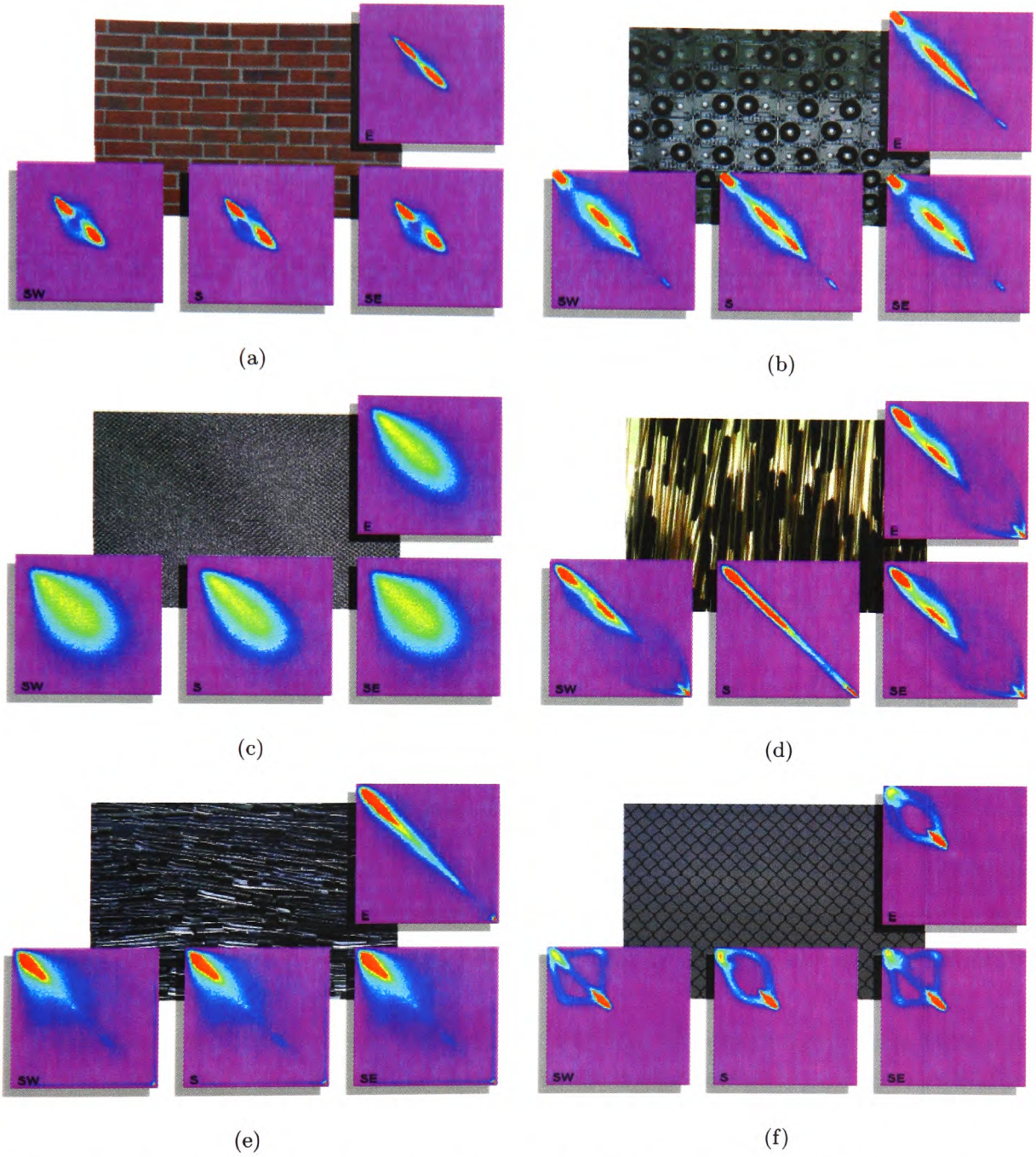


Figure 41: Cooccurrence matrices East, South-East, South, and South-West being applied to texture images 114.6, 114.11, 114.12, 114.37, 114.47, and 114.49.

In most images the neighbouring pixels are fairly similar for most pixel-pairs. The according cooccurrence values are on or near the main diagonal, the higher (lighter) the values are, the further away from the origin[13]. Values that are off the main diagonal represent contrast in the image.

By testing all surrounding neighbours[14] high or low contrast in the horizontal, vertical, and both diagonal directions is being detected.

Usually the CMs are being used directly on the texture image, just as shown in Fig. 41. But the Texture Classification System provides the possibility to easily use the CM on filtered or transformed images. An application of the cooccurrence matrix on the high-pass and low-pass residue images produced by the WT has been tested with good results. Using the LAWS micro statistics, with or without the TEM, is useful, too. Those transformations enhance edges, which means in the case of texture images, that the image contrast is being enhanced. This again can be detected and statistically evaluated by the CM.

It should be mentioned that such combinations have, to the author's knowledge, not been used by anyone before.

More information about cooccurrence matrices can be found in [CJ96a, CJ96b, EP93, KP96, Sha93].

2.5.2.2 Wide-Cooccurrence Matrices

Another new idea of the author concerning the CMs is the introduction of the Wide Cooccurrence Matrix (WCM). The difference to the CM is that not only the neighbouring pixels are being used (cf. Fig. 42), but also neighbours that are two (cf. Fig. 43) or three (cf. Fig. 44) pixels away.

By this means changes in an image can be detected, that are of high contrast but low frequency. This is important for texture images, as texture elements are not necessarily only two pixels wide.

Additionally the direction dependency of texture features can be detected in greater detail.

[13]The origin is, as usual for images, in the top left corner.

[14]The left-hand side neighbour of pixel $p_{i,j}$ would give the to the main diagonal mirrored cooccurrence value and thus reduce the information.

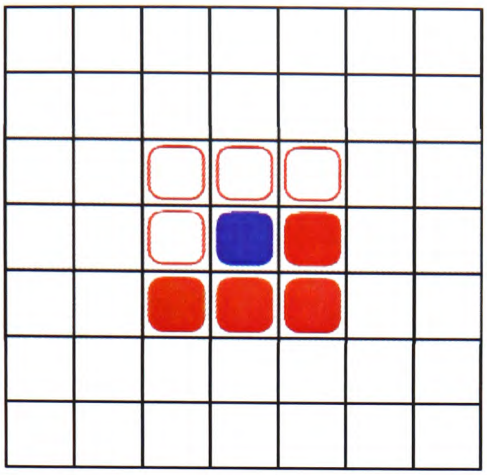


Figure 42: Cooccurrence matrices 4 neighbours

For the Texture Classification System the WCM shown in Figs. 42, 43, and 44 were used. The operator allows neighbours up to twenty pixels away.

2.5.2.3 Multi-Dimensional Cooccurrence Matrices

The next new idea is using a Multi-Dimensional Cooccurrence Matrix (MDCM). Whilst the CM as well as the WCM are two-dimensional, the MDCM actually tests more neighbours at once. Fig. 45 shows as an example a four neighbour MDCM which produces a five-dimensional WCM.

The number of possible different MDCMs is enormous. While the two-dimensional already produces 24 CMs, a three-dimensional approach using the WCM as a basis can bring forth up to 552 matrices[15]. The number off possible matrices is:

$$n_D = \prod_{i=n_{\mathrm{WCM_{max}}}-D+2}^{n_{\mathrm{WCM_{max}}}} i \tag{36}$$

with D being the dimension, equal to the number of neighbours plus one, and $n_{\mathrm{WCM_{max}}}$ being the number of possible neighbours. The example in Fig. 45 would be one of 255.024 possibilities[16] for that dimension.

It is clear that this approach is very specialised and will only give good results, if either the computing power is so great, that calculating all the matrices is not a problem

[15] 24×23
[16] $24 \times 23 \times 22 \times 21$

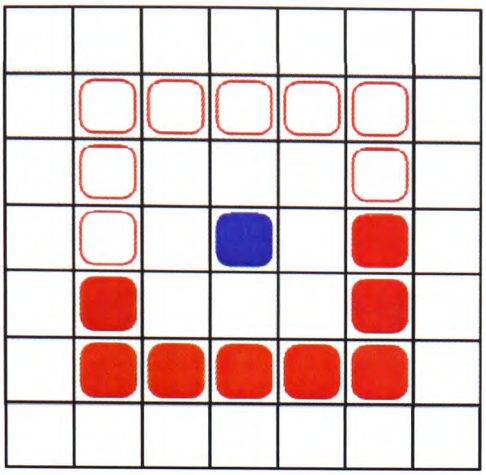


Figure 43: Cooccurrence matrices 8 neighbours

concerning time and amount of data, or special subsets are being used. Finding good subsets is a task for the Genetic Algorithm (GA)-module described in Section 2.8.

It is also important to note, that the number of image pixels must large enough to obtain a MDCM that can be statistically evaluated with good results.

The MDCM has not yet been implemented for the Texture Classification System. At the moment it is just a theoretical approach.

2.5.2.4 Statistics on Cooccurrence Matrices

As already mentioned in Section 2.5.2.1 the statistical features of the CMs are of interest for the Texture Classification System.[17] The WCM-module produces values for the energy (Eqn. 37), the contrast (Eqn. 38), the entropy (Eqn. 39), and the correlation (Eqn. 40).

Energy:

$$T_1(\delta) = E_d(\delta) = \sum_{i=1}^{Q}\sum_{j=1}^{Q} C_\delta(i,j)^2 \tag{37}$$

Contrast:

$$T_2(\delta) = K_d(\delta) = \sum_{i=1}^{Q}\sum_{j=1}^{Q} (i-j)^2 C_\delta(i,j) \tag{38}$$

Entropy:

$$T_3(\delta) = H_d(\delta) = -\sum_{i=1}^{Q}\sum_{j=1}^{Q} C_\delta(i,j)\log C_\delta(i,j) \tag{39}$$

[17]The images shown in Fig. 41 have only been produced for this book. For the calculation of features only the integer valued matrices are of interest.

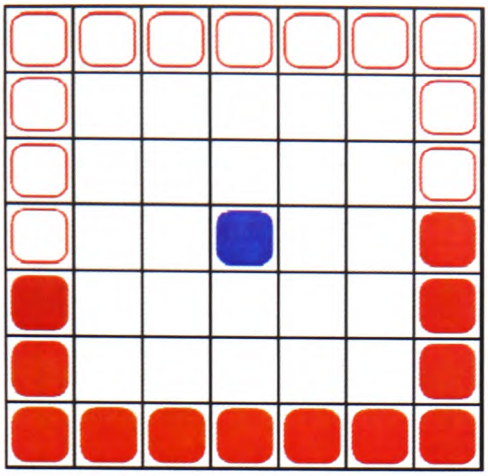


Figure 44: Cooccurrence matrices 12 neighbours

Correlation:

$$T_4(\delta) = \frac{1}{\sigma_m \sigma_n}(\sum_{i=1}^{Q}\sum_{j=1}^{Q} ijC_\delta(i,j) - \mu_m\mu_n) \tag{40}$$

with μ_m, μ_n, σ_m, σ_n being the average and root mean square (rms) of p_m and p_n:

$$p_m(i) = \sum_{j=1}^{Q} C_\delta(i,j) \quad \text{and} \quad p_n(j) = \sum_{i=1}^{Q} C_\delta(i,j) \tag{41}$$

When all the WCMs are being calculated this alone gives a feature vector of size 96 for each input image, where every image is a filtered or/and transformed subimage of the original texture image.

2.5.2.5 Logarithmic Cooccurrence Matrices

A further new idea of the author is to logarithmise the cooccurrence matrices before they are statistically scrutinised. This reduces the influence of the majority values, usually those of neighbours of near equal greylevel values.

There are many different ways how to transform the Logarithmic Cooccurrence Matrix (LCM). Which is the most appropriate approach depends on the type of texture images, or, if used in a different application, the characteristics of the images in question.

Possible transformation functions that were tested are:

$$C_{L_2} = \log_2 C \tag{42}$$

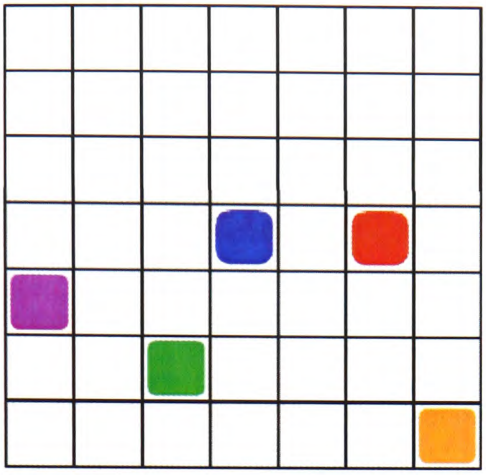


Figure 45: Cooccurrence multi-dimensional neighbours

$$C_{L_2^2} = \log_2 C^2 \tag{43}$$

$$C_{L_e} = \ln C \tag{44}$$

$$C_{L_{10}} = \log_{10} C \tag{45}$$

with C being the original cooccurrence matrix and C_L the resulting logarithmised matrix.

2.6 Clustering and Fuzzy Clustering

2.6.1 Sharp Clustering

All the texture features extracted by the statistical modules described in Section 2.5 are combined into a feature vector containing n elements. This is one data point in an n dimensional plane. All the feature vectors of the textures build non-overlapping[18] groups, or clusters, of points in the n^{th} dimension. Fig. 46 gives an idea for three dimensions.

The points within a cluster are 'more similar' to one another than to points of other clusters. The term 'more similar' means in this context closer by some measure of proximity. This measure can be e. g. the L_p distance:

$$D^p(x_k, x_l) = \left(\sum_{i=1}^{n} |x_{ki} - x_{li}|^p \right)^{\frac{1}{p}} \quad \forall\, p > 0 \quad . \tag{46}$$

[18] If the clusters overlap, sharp clustering cannot be applied. A solution to this problem is using fuzzy clustering, discussed in Section 2.6.2.

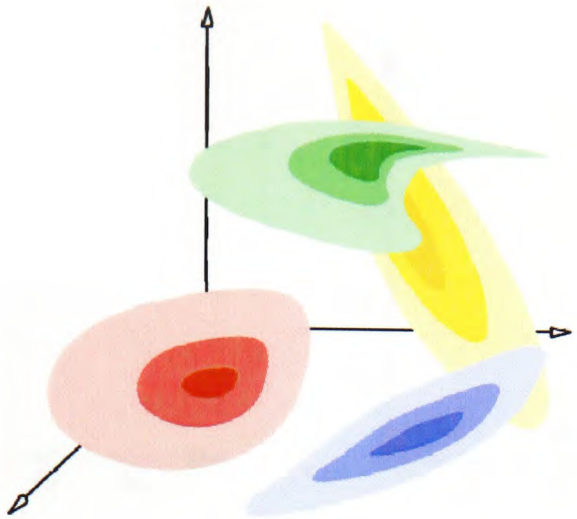


Figure 46: Cluster in a 3D plane.

For $p = 1$ this is the Hamming distance, for $p = 2$ the Euclidean distance. Weighing the L_p distance will give the Minkowski distance:

$$D^{\tilde{p}}(x_k, x_l) = \left(\sum_{i=1}^{n} w_i |x_{ki} - x_{li}|^p \right)^{\frac{1}{p}} \quad \forall\, p > 0 \quad . \tag{47}$$

Every feature vector of a data-set partitioned in the above mentioned way is assigned to exactly one cluster. Each cluster can be referenced by a single reference point, usually the mean value of all members of this cluster.

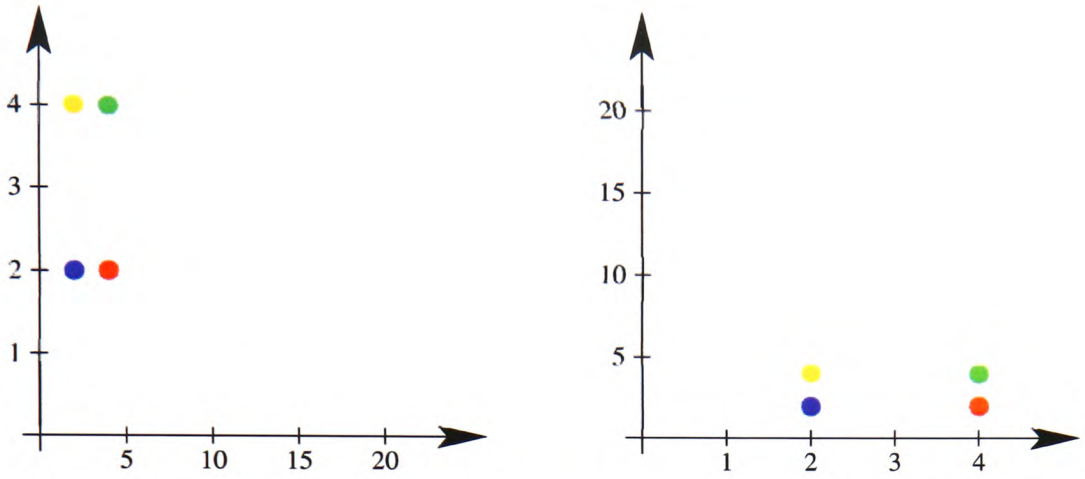
The scaling of the values in the feature vector is very important for a correct classification. Fig. 47 shows two different results for identical unscaled data.

There are two main groups of cluster analysis methods. One is hierarchical, the other non-hierarchical. The hierarchical method divides the data-set into two most dissimilar sub-groups. Those sub-groups are again divided by the same method, and so on. The non-hierarchical method assigns each element of the data-set sequentially to one cluster.

2.6.1.1 The k-means Algorithm

The k-means algorithm first described by MACQUEEN [Mac67] is a non-hierarchical method.

The number of clusters k is known beforehand. The first step is to set k starting points in the n dimensional plane. Those starting points form the cluster centres at the



(a) The obvious clusters are blue-red and yellow-green.

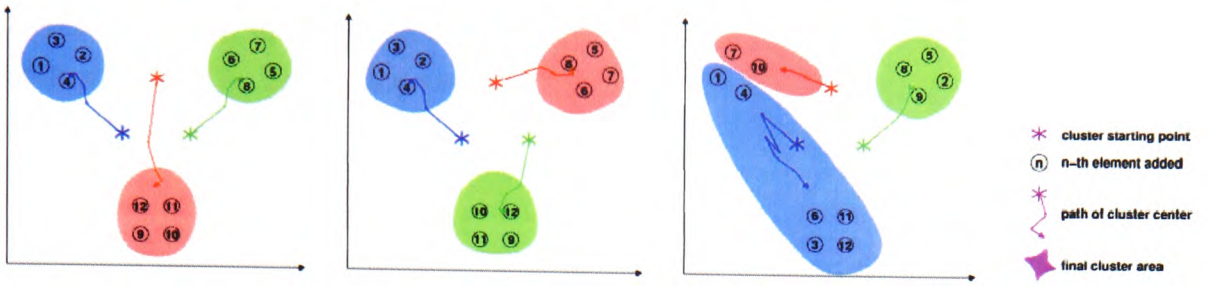
(b) With a different scaling the clusters are now blue-yellow and red-green.

Figure 47: The clustering result is dependent on the scale of the incoming data. The examples show the points (2;2), (2;4), (4;2), and (4;4).

beginning. The points can be chosen by random or, and this is more effective, from the first k feature vectors that are known to be dissimilar. The next step is to allocate the following feature vectors to one of the starting clusters. The cluster which is closest to the new element is chosen. With each new element the cluster centre is recalculated. Once all elements of the data-set have been allocated, the final cluster centres are used as fixed centres for a second round. All elements are allocated again, but this time the cluster centres are not recalculated.

The algorithm steps in short:

1. Define k, the number of clusters.
2. Initialise the clusters by random or by choosing selected elements.
3. Sequentially assign each element of the data-set to the closest cluster centre. Calculating the distance can be done by different methods.
4. After every new element recalculate the centre of the clusters, usually by calculating the centre of gravity for each cluster.
5. The final cluster centres are used for a second pass. This time the centres stay fixed.

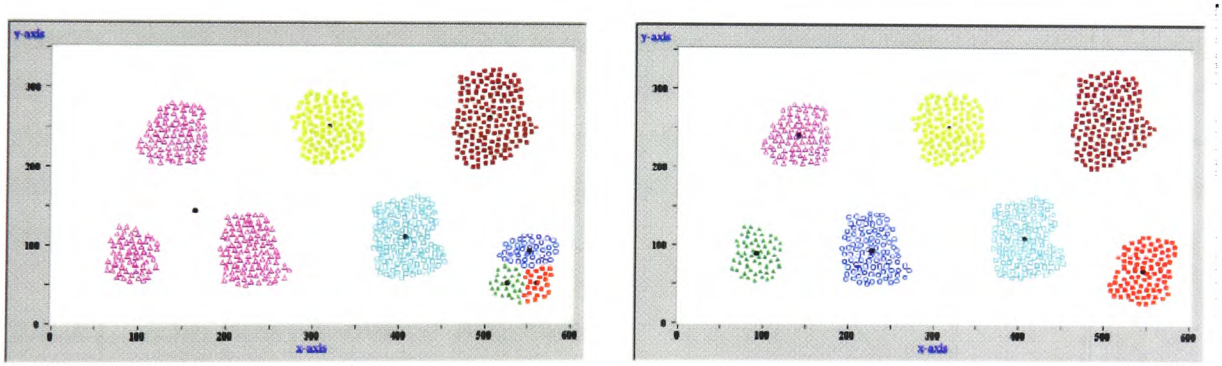


(a) (b) (c)

Figure 48: The movement of cluster centres during the first assignment of data elements depends on the order of the incoming data.

This algorithm has certain limitations. A variation is to rerun the algorithm until convergence is reached.

Setting the initial starting points can be important. Fig. 49 shows two different, but mathematically correct classification results. Feature vectors of known quality provide good starting points.



(a) (b)

Figure 49: The classification result, good or bad, depends on the initial starting points.

The method of unsupervised classification into k groups works well for texture classification, as the number of textures that have to be trained is known a priori.

If k is not constant a metric has to be used that finds the optimal number of clusters. Therefore the quality of a classification has to be judged. Literature describes a number

of different methods [Rom84, McQ87, KR90]. Using the Automatic-Optimisation-System (AOS) described in Section 3.4 is a possible solution for the problem to determine the optimal size of k. So far this has not been tested for the Texture Classification System, as the number of textures which are used is known.

2.6.2 Fuzzy Clustering

A problem for sharp clustering is to handle data points that are close to more than one cluster. The Fig. 50 showing the 'butterfly problem' gives a graphical example . The central data point in Fig. 50(a) should belong equally to both clusters as shown in 50(c) and not to only one as it is the case when sharp clusters are applied as in figure 50(b). In the case of the butterfly problem the clusters have to overlap.

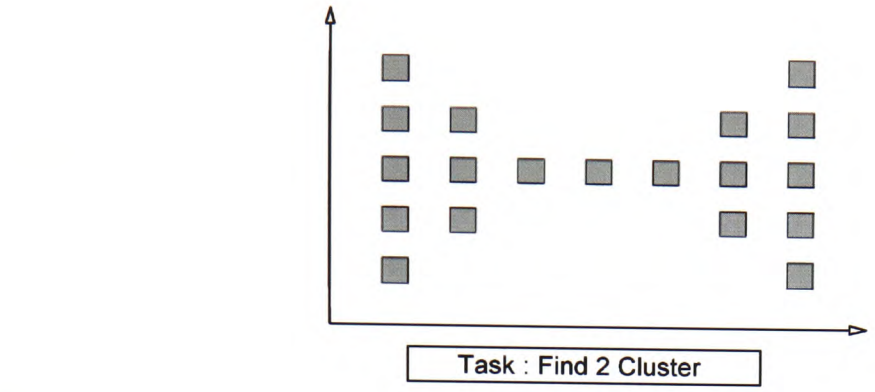
With increasing numbers of dissimilar textures the chance of clusters overlapping in one or more dimension increases. Therefore new methods have to be adopted to overcome this problem.

Fuzzy logic methods have a potential for handling uncertain knowledge [Sto93, Zad65, Zim91]. Thus it is possible to classify texture of which the feature vectors do not point to the centre of a certain cluster but into an overlapping area of two or more clusters [Bez73, Dun73, GG89, Til93]. Such a texture has a membership value of a certain height for every cluster of the plane. Usually this value equals zero for almost all clusters and obtains a high value for the cluster describing the texture in question. Texture images that have membership values of about equal height for two or more clusters have to be treated in a postprocessing system.

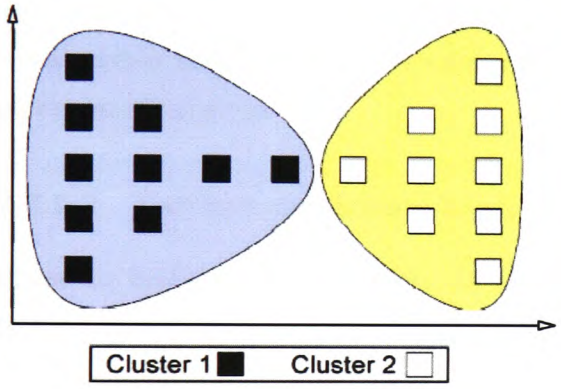
2.6.2.1 The fuzzy-c-means Algorithm

One possible fuzzy clustering method is to use the fuzzy c-means algorithm [Bez73, DH73, Dun73]. This is probably the best known of all fuzzy clustering methods and the base for more specialised versions like the fuzzy c-varieties [Bez81] which uses n dimensional prototypes, the adaptive fuzzy clustering [Dav89] for elliptical cluster shapes and the fuzzy c-spherical shells [Kri93] for hyperspherical shells to name but a few.

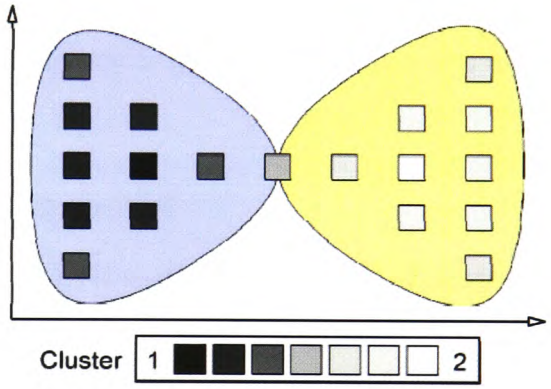
The fuzzy c-means algorithm is the fuzzy extension of the sharp cluster analysis. As before the distances between all m feature vectors of the training set and the cluster centres



(a) Incoming data



(b) Sharp clustering



(c) Fuzzy clustering

Figure 50: The butterfly problem.

v have to be minimised. The difference is that every feature vector gets a membership value μ for every cluster c, noted in the fuzzy partitioning matrix U.

The minimisation of the function

$$\min z_w(U, v) = \sum_{k=1}^{m} \sum_{i=1}^{c} \mu_{ik} D_{ik}^{p} \tag{48}$$

with w being a weighing or fuzzification factor has to obey the supplementary conditions

$$\sum_{i=1}^{c} \mu_{ik} = 1 \quad \forall\, 1 \leq k \leq m \quad , \tag{49}$$

$$\mu_{ik} \in [0, 1] \quad \forall\, 1 \leq i \leq c, 1 \leq k \leq m \quad , \tag{50}$$

$$0 < \sum_{k=1}^{m} \mu_{ik} < m \quad . \tag{51}$$

Finding the optimal fuzzy partition matrix follows the same routine as in the sharp clustering algorithm. After initialisation of the number of clusters and the fuzzification factor, the initial cluster centres are determined and the membership values calculated and thus the fuzzy partitioning matrix established. The cluster centres will then be repositioned according to the partitioning matrix and new membership values are calculated. This will continue until the difference between old and new partitioning matrix is small enough.

The problem with this algorithm is of course that the distance measure promotes only certain kinds of compact clusters, namely hyperspherical clusters. Secondly the membership values for feature vectors that are far away from all cluster centres is about equal for all clusters.

2.6.2.2 Further specialised fuzzy clustering algorithms

Since the first development of the fuzzy-c-means (FCM) algorithm a number of fuzzy clustering algorithms have been designed. In principle they are based on the FCM algorithm.

Most prominent are the algorithms described by DONALD E. GUSTAFSON and WILLIAM C. KESSEL [GK79] and the one by ISAK GATH and AMIR B. GEVA [GG89]. They both have the ability to find hyperelliptical cluster which is not possible for the fuzzy-c-means.

The main difference is the way the distance is calculated. Whilst the FCM algorithm uses the Euclidean distance, the Gustafson-Kessel (GK) algorithm takes the Mahalanobis distance:

$$d_{ik}^2 = (x_k - v_i)^T A_i (x_k - v_i) \tag{52}$$

with the matrices A_i given by:

$$A_i = \sqrt[p]{\det(S_i)} S_i^{-1} \tag{53}$$

and the fuzzy covariance matrix:

$$S_i = \frac{\sum_{j=1}^{n} \mu_{ij}^m (x_j - v_i)(x_j - v_i)^T}{\sum_{j=1}^{n} \mu_{ij}^m} \tag{54}$$

To calculate S^{-1} the number of elements n must be greater than the dimension, i. e. the number of features i in an element.

The next variant, the Gath-Geva (GG) method, also known as Gaussian mixture decomposition (GMD), uses the Gaussian distance. This is given by:

$$d_{ik}^2 = \frac{\sqrt{\det(S_i)}}{P_i} e^{(\frac{1}{2}(x_k - v_i)^T A_i (x_k - v_i))} \tag{55}$$

with the matrices A_i given by:

$$A_i = S_i^{-1} \tag{56}$$

and the probability factor P_i:

$$P_i = \frac{\sum_{j=1}^{n} \mu_{ij}^m}{\sum_{j=1}^{n} \sum_{k=1}^{c} \mu_{ij}^m} \tag{57}$$

The factor P_i influences the size of a cluster.

Usually another clustering method, e. g. k-means or fuzzy-c-means, is used to initialise the partition matrix for the GG algorithm.

There are further but less well known methods. They only shall be mentioned by their name and special clustering abilities in Table 11.

FCM	fuzzy-c-means	spherical cluster of approx. the same size
GK	Gustafson-Kessel	ellipsoidal cluster of approx. the same size
GG	Gath-Geva	ellipsoidal cluster of varying size
GMD	Gaussian mixture decomposition	different name for the GG algorithm
FCV	fuzzy-c-varieties	detection of linear manifolds
AFC	adaptive fuzzy-c-varieties	detection of line segments
FCS	fuzzy-c-shells	detection of circles
FCSS	fuzzy-c-spherical-shells	detection of circles
FCR	fuzzy-c-rings	detection of circles
FCQS	fuzzy-c-quadratic-shells	detection of ellipsoids
FCRS	fuzzy-c-rectangular-shells	detection of rectangles

Table 11: Fuzzy clustering methods and their special abilities according to [HKK97].

2.7 Neural Networks

2.7.1 Analogy to the human brain

The human brain and its exact functionality is still not fully understood by researchers. Yet, some aspects of this 'biological processor' are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, does not appear to regenerate.[19] These cells, of which there are approximately 10^{11}, are known as neurons. Each of these neurons can connect with up to 200,000 other neurons, although 1,000 to 10,000 is typical. The power of the human mind comes from the sheer numbers of these basic components and the multiple connections between them and from the ability to learn by training.

The individual neuron is a complex system in which information is conveyed via a host of electrochemical pathways. There are over one hundred different classes of neurons. Together these neurons and their connections form a process which is not binary, not stable, and not synchronous. In other words, it is quite different to the currently available VON NEUMANN type computers, and even to artificial neural networks. These artificial neural networks try to replicate only the basic idea. Compared to real neurons the artificial ones are quite primitive. But for the software engineer who is trying to solve problems, neural computing was never about replicating a human brain. It is about a new way of solving problems. In Table 12 are some important characteristics of human neural networks summarised.

2.7.2 Description of a Neural Network

Artificial Neural Network (ANN) are relatively simple electronic or algorithmic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by a human brain.

The biologically inspired methods of computing are one of the major advancement in computing[20]. Even the brains of simple animals are capable of functions that are currently

[19]Latest research suggests that regeneration of brain cells is possible, but in the mean those cells exist as long as the organism itself.

[20]Not only Neural Network (NN), but Fuzzy Logic (FL) and Evolutionary Computing (EC), too.

number of neurons	$> 10^{11}$
number of connections (synapses)	$> 10^{14}$
number of afferent (input)	10 %
number of efferent	90 %
storage capacity	equiv. to $10^{13} - 10^{15}$ bits
average brain weight	1.4 kg
average neuron weight	1.210^{-9} g
signal propagation rate	5 - 125 ms^{-1}
synapse traverse time	1 ms
refractory period	10^{-2} s
firing frequency	50 - 100 spikes/s
synapses (general types)	excitatory and inhibitory
membrane potential	triggers firing
operation mode	asynchronous (some rhythm imposed)

Table 12: Characteristics of human neural network.

impossible for computers. Computers do routine things well, like data storing and sorting or performing complex numerical calculations. But normal computer algorithms are not capable of generalising patterns recognised in the past into actions of the future.

Advances in biological research give an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. This process of storing information as patterns, utilising those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilise traditional algorithmic programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems.

The field of neural computing also utilises words very different from traditional computing, words like behave, react, self-organise, learn, generalise, and forget.

2.7.3 Basic Concepts in Neural Computing

The fundamental processing element of a neural network is a neuron. This building block incorporates a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in a specific way, performs a generally nonlinear operation on the result, and then outputs the final result. Fig. 51 shows a simple neuron. Within humans there are many variations on this basic type of neuron. Yet, all natural neurons have the same four basic components. These components are known by their biological

names – dendrites, soma, axon, and synapses. Dendrites are hair-like extensions of the soma which act as input channels. These input channels receive their input through the synapses of other neurons. The soma then processes these incoming signals over time and turns that processed value into an output which is sent out to other neurons through the axon and the synapses.

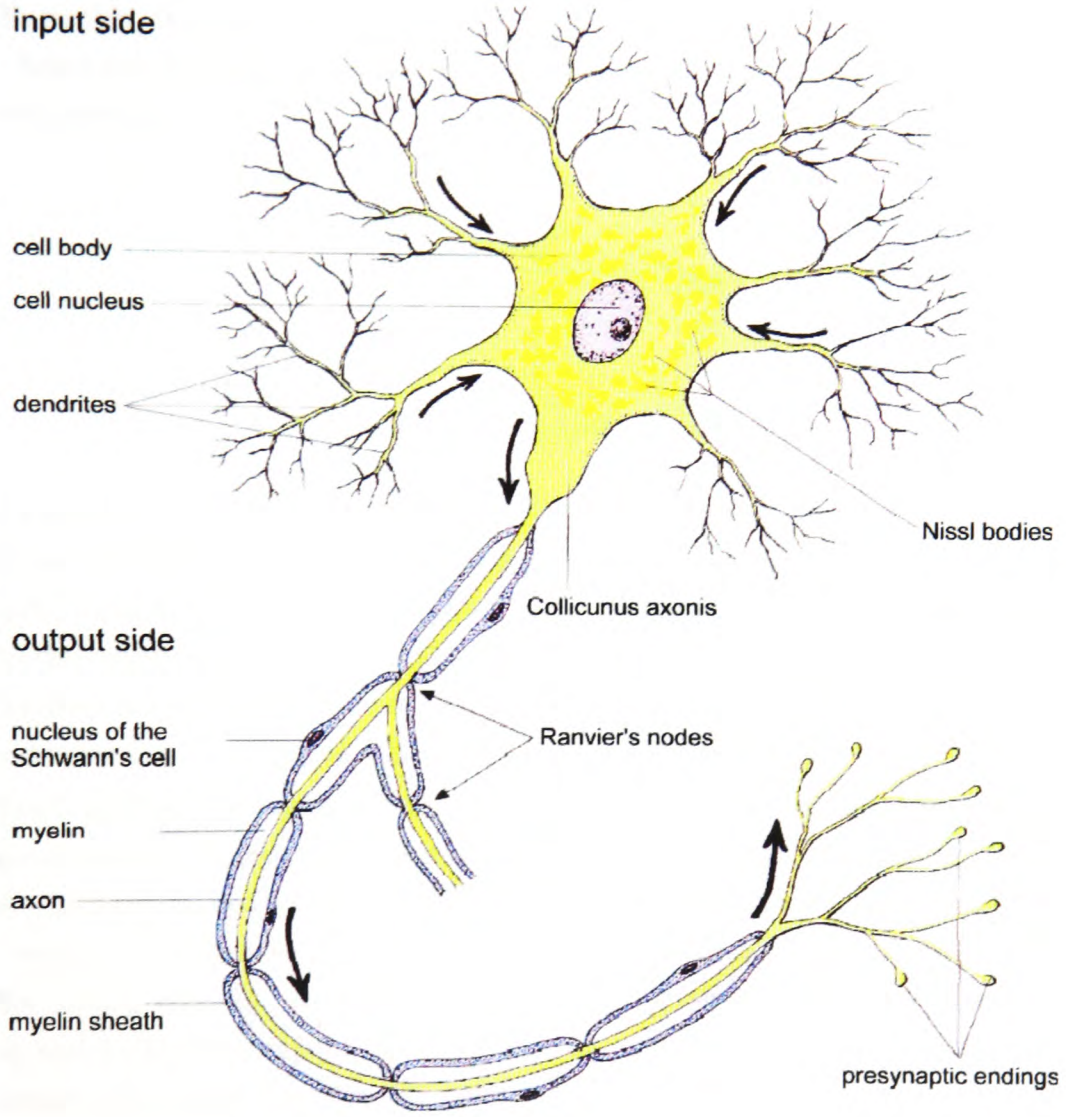


Figure 51: Drawing of a simple neuron. (adapted from [SM02])

Recent experimental data has provided further evidence that biological neurons are structurally more complex than the simplistic explanation above. They are significantly more complex than the models of artificial neurons known and used today.

Currently the goal of artificial neural network research is not the recreation of the brain but to better understand nature's capabilities. With this knowledge solutions for problems can be engineered that have not been solved by traditional computing.

To do this, the basic unit of neural networks, the artificial neurons, simulate the four basic functions of natural neurons. Fig. 52 shows a fundamental representation of an artificial neuron.

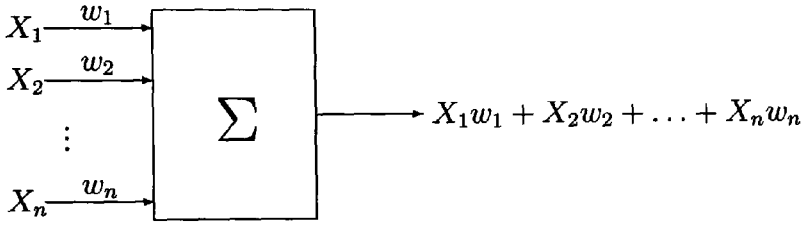


Figure 52: A basic artificial neuron.

As shown in Fig. 52, inputs to the network are represented by X_n. Each of these inputs are multiplied by a connection weight w_n. In the simplest case, these products are simply summed, processed through a transfer function to generate a result, and then sent to the output. This process lends itself to physical implementation on a large scale. This hardware implementation is still possible with other network structures which utilise different summing functions as well as different transfer functions.

Some applications require binary answers, e. g. the recognition of text or the identification of speech. These applications are required to turn real-world inputs into discrete values, which are limited to a known set, e. g. the ASCII character set or the most common 5,000 words of a language. Due to this limitation of output options neurons are needed that not simply sum up, and thereby smooth inputs, but perform binary properties of OR-ing and AND-ing of inputs. These functions, and many others, can be built into the summation and transfer functions of a network.

For other applications different output behaviour is needed. Some machines need continuous valued parameters, e. g. robots, or input data integration over the time is needed, produced by time dependent networks.

A backpropagation network is created by complying to the following three rules	
1.	When the complexity in the relationship between the input data and the desired output increases, the number of processing elements in the hidden layer should also increase.
2.	If the process which is being modelled is separable into multiple stages, additional hidden layers may be required. If the process is not separable into stages, additional layers may simply enable memorisation and not a generalised solution.
3.	The amount of training and testing data sets available defines an upper bound for the number of processing elements in the hidden layers. To calculate this upper bound, the number of input/output pairs in the training set is divided by the total number of input and output processing elements in the network times a scaling factor. This scaling factor is between five and ten. Larger scaling factors of e. g. twenty are used for relatively noisy data, smaller ones, e. g. two or three, for very clean input data that has an exact relationship to the output. It is important that the hidden layers have relatively few processing elements, as too many neurons will lead to the memorisation of the training set.

Table 13: Design rules for artificial neural networks.

2.7.4 Texture Identification with Neural Networks

In this section the use of Neural Networks (NNs) for texture classification shall be described. Additionally some NNs shall be named that are well suited for an implementation within the Texture Classification System. Most of these networks are available through the SNNS software package, described in Section D.1.

2.7.4.1 Employment of Neural Networks for Texture Classification

The NNs can be used in two different ways. The first approach is similar to the employment of clustering or fuzzy clustering. A statistical feature vectors is fed into an appropriate network. This network, after it has been trained and tested with sufficient data, classifies the statistical data and thus the texture images. The feature vector is supposed to be not too large. As the design rules in Table 13 on page 75 have to be obeyed, NNs can easily become very large.

The second approach is to train a suitable network directly with image data. Each pixel of a preprocessed, filtered or transformed image is connected to one input layer neuron. This means that the texture features are not explicitly used but implicitly hidden

in the texture image. It is clear that either the images must be rather small concerning the number of pixels or the network sufficiently big. With the computational resources available in a normal laboratory using networks with very many neurons is not yet possible. On the other hand, choosing very small input images reduces the chance of classifying a larger group of textures considerably.

The number of possible ways how to incorporate NNs as classification modules is almost limitless. Next to the choice of network the optimal number of input, output and hidden layer neurons has to be established for the specific application. In the future, when computer hardware allows larger NNs and self optimisation, hybrid systems could be used. Hybrid Neural Networks (HNNs) are by the author's definition[21] either systems of different type NNs used sequentially and/or in parallel (high-level HNNs) or NNs which are composed of different type neurons within one network (low-level HNNs). These ideas have, due to the mentioned hardware limitations, not been tested for texture classification. To the author's knowledge HNNs have so far not been implemented or described for other tasks, too.

The next sections give a condensed overview over existing types of NNs that can be employed for texture classification.

2.7.4.2 Learning Vector Quantisation

The Learning Vector Quantisation (LVQ) NNs are single layer networks. They can be implemented with an additional layer of input neurons, a so called Kohonen layer, which has no preprocessing function. The weights are adjusted by the input activity and the class information. A number of different learning strategies exist (LVQ1, LVQ2.1, LVQ3, OLVQ1). [Koh89, KKLT92]

The LVQ is well suited for classification tasks but needs class information (i. e. which input vector belongs to which class) for the training.

2.7.4.3 Self-Organised Maps

The Self-Organised Maps (SOM) NNs are single layer networks just like the LVQ. They are implemented under the consideration of the neighbourhood relations. The weights are

[21] In literature the phrase Hybrid Neural Network is also used for NNs which are part of a classical system using symbolic representation. Additionally in neurobiology the phrase is used for neural networks consisting of living nerve cells interacting with model neurons.

adjusted by unsupervised learning which holds the danger of uncorrectable topological map defects. [Koh82, Koh84, Koh90b, KKL92]

Like the LVQ the SOM is well suited for classification tasks.

2.7.4.4 Adaptive Resonance Theory Neural Network

As an alternative to the before mentioned network types the NNs of the Adaptive Resonance Theory (ART)-family are very interesting for classification tasks in a natural surrounding. These NNs try to solve the stability-plasticibility-problem (i. e. how to train new associations without loosing old ones). The advantage is that the ART NNs are retrainable, the disadvantage that they are mathematically rather complex, which means computationally demanding. [CG88, Kow95, RG96]

ART-1

The ART-1 network consists of two layers of neurons, the F1-layer for testing and the F2-layer for recognition. Only binary input vectors are allowed. A non-parallel implementation is rather slow but nevertheless well suited for classification tasks. [Gro76]

ART-2

The ART-2 network can handle continuous input vectors. Therefore the network structure had to be expanded by three additional layers. The F1-layer consists now of four layers with feedback mechanisms and six different types of neurons. [CG87a]

It is, like the ART-1 NN suited for classification but rather slow and complex.

ART-2A

The ART-2A is an optimised version of the ART-2 NN. The convergence criteria are reached two to three powers of ten faster due to simplified equations. [CGR91b]

ART-3

The ART-3 model is a very complex model based on the ART-2 layout. Asynchronous inputs can be used and the chemical properties of real synapses are emulated. From the technical point of view this NN is too complex, from the biological point of view too simple. If someone succeeds to implement ART-3 on a very fast hardware it will probably be well suited for classification tasks. At the moment it is purely academic. [CG87b]

ARTMAP

The ARTMAP is a combination of two ART networks glued together head on. The input vector is fed into the F1-layer of the first ART network, and the corresponding output vector into the F1-layer of the second ART network. Both F2-layers are connected via a so called MAP-field, which is a weight matrix. Thus a supervised learning is obtained and after the training period, which is rather slow, the MAP-field helps to accelerate the classification. [CGR91a]

Fuzzy ART and Fuzzy ARTMAP

Fuzzy ART is a combination of Fuzzy Logic (FL) with ART-1. By this way continuous input vectors can be used together with the ART-1 network. [BE96, CGR91c, WSS94]

This idea has been utilised for an implementation of a Fuzzy ARTMAP NN. This is the latest member of the ART family. [CGM+92, MH95, RP00]

2.7.4.5 Other Network Models

There are quite a number of other NN and Fuzzy Neural Networks (FNN) models that could be used for classification tasks. Some of these are Fuzzy Kohonen Networks [Bro95], Radial Basis Function (RBF) and Cubic Basis Function (CBF) as well as Fuzzy RBF and Fuzzy CBF NNs [Har95b, HPG94], Cascade Correlation (CC) NNs [Jan94, LFJ94] and FNNs after LIN and LEE [Har95a], PAL and MITRA [Kli95] or HAYASHI and ISHIBUCHI [vV95].

More general articles and good overviews on NNs can be found in [Roj93, Sch93a, SHG90, Zel94] to name but a few.

2.8 Genetic Algorithms

2.8.1 General Introduction

A GA has the capability of finding very good local or even global optimal solutions in complex data-planes (cf. [SHF94, SD98]). Every feature[22] is associated to one *gene*—a boolean element—and all genes compose the equivalent of a *Desoxyribonucleic Acid (DNA)*. If the gene is set to zero the associated feature is not applied[23] and it is used if the gene is set to one.

To begin with a starting population is created. This means a number of DNA-strings (or DNA-vectors) are build with the gene-elements randomly set to one or zero. This is the first half of the starting population, the other half consists of the negated first half. The negation provides an even distribution of zeros and ones for every gene throughout the starting population. This is quite important as usually the number of created DNA-strings is small against the size of the individual strings. Therefore it could easily happen, that some of the gene-elements are set to either zero or one for all members of the starting population.[24]

In the next step all or some members of the population, called 'parents', are used to create a new generation of members, called 'children' or 'offspring', by exchanging parts of the DNA-string. This is known as *crossover* (cf. Section 2.8.2). Depending on the way of selection and production of DNAs the population can grow rapidly. Additionally some DNAs can be mutated to avoid getting stuck in a local optimum (cf. Section 2.8.3).

2.8.2 Crossover

By the means of crossover the new members of the population are being created. This technique is copied from nature, where crossover and selection results in better adopted creatures in following generations. This procedure is known as evolution and was first described by DARWIN.

The technical realization has brought forth different solutions like the single-cut, dual-cut and shuffle-cut crossover as described below. Further methods are the partially

[22] A feature can e. g. be an element of a feature vector, or part of a parameter set for an operator.

[23] I. e. not used as a feature value in the following modules of the Texture Classification System, or not used as a parameter for an operator setup.

[24] The probability is almost 10% for a 100-DNA-element-sized starting population of 10 members.

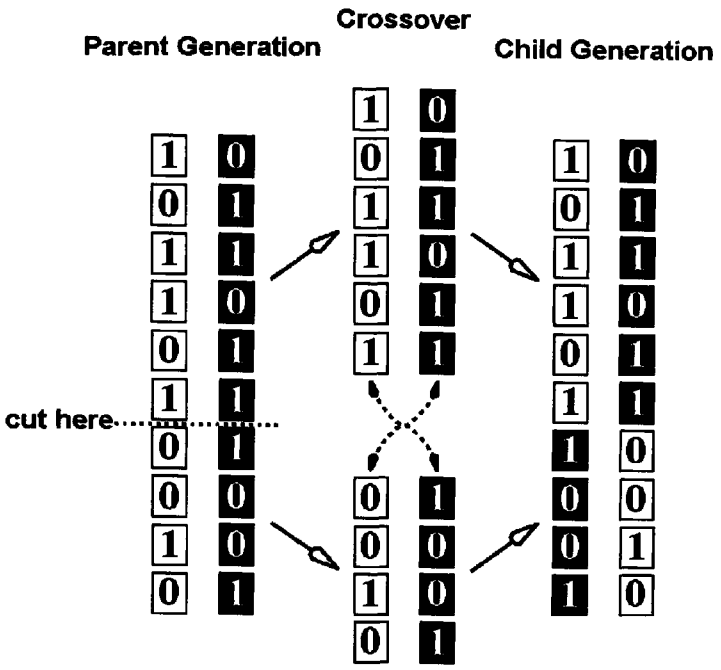


Figure 53: Single-cut crossover.

mixed crossover and the uniform order-based crossover. As those two methods are not useful for the Texture Classification System they will not be described. See also [Dav91, Gol89, SMM+91].

2.8.2.1 Single Cut

Fig. 53 shows graphically how the single-cut crossover works. The spot where the DNA-string is cut is chosen by random. The parts below the cut are exchanged between the two mating members of the population, thus creating two new members, the children. This crossover technique is also called one-point crossover.

2.8.2.2 Dual Cut

Actually the single-cut crossover is a special dual-cut crossover with the second cut always at the end of the DNA-string. The general version of the dual-cut crossover is shown in Fig. 54. This crossover variant avoids that the first and last elements are 'torn apart' at every iteration.

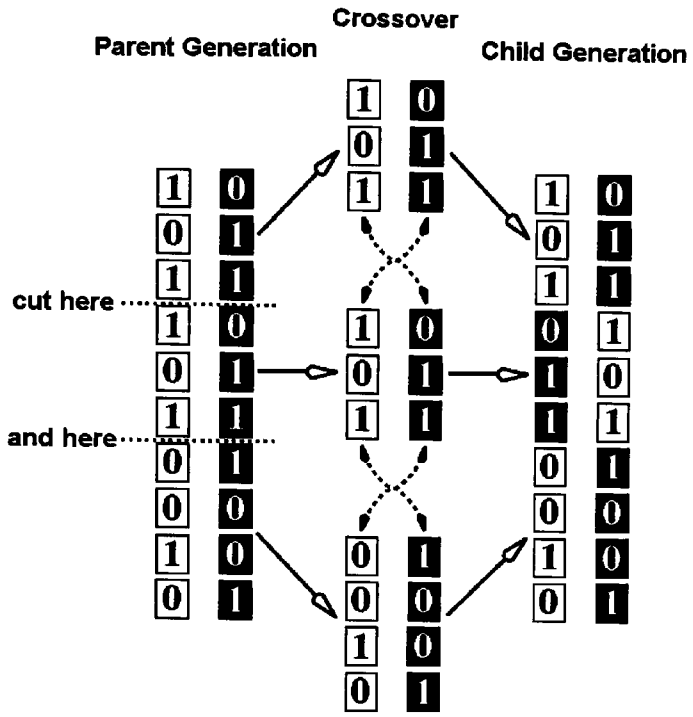


Figure 54: Dual-cut crossover.

The dual-cut crossover, which also is called two-point crossover, promotes the idea of a ring-DNA which has no first or last element (cf. Fig. 55). The positions of all genes are equal.

For the technical implementation this is of no importance, but it helps to understand the differences between the single-cut and dual-cut crossover techniques.

A problem with the single-cut crossover is that only certain children can be produced by mating. An example will show the problem. Assume $P_1 = \{0, 1, 0, 1, 1, 0\}$ and $P_2 = \{1, 1, 0, 0, 1, 1\}$ are the parents. The equal values can be disregarded, therefore the search space in this example is three-dimensional. In general, the two parents represent the opposite corners of an n-dimensional hyper-cube, where n is the dimensionality of the search space for these parents (the number of genes in which the DNA-strings differ). There are $2^n - 2$ other corners in this hyper-cube that represent $2^{n-1} - 1$ complimentary pairs of children. For the example the children pairs (C_1, C_2), (C_3, C_4), and (C_5, C_6). Fig. 56 shows it graphically.

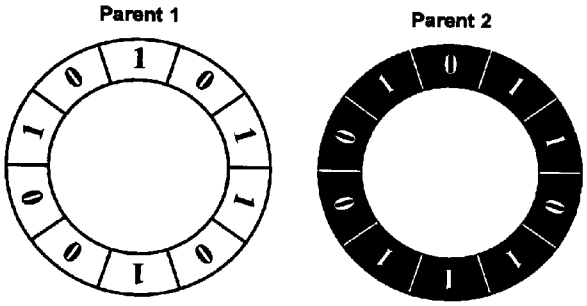


Figure 55: Ring-DNA.

If a single-cut crossover is used on these parents, only the pairs (C_1, C_2) and (C_3, C_4) can be generated with equal probabilities. The children (C_5, C_6) cannot be generated directly, only as grandchildren. If a dual-cut crossover is used, all three complementary pairs can be generated with a 25% probability. The missing 25% are clones, i. e. children that are identical to the parents. This happens, if the random values for the first and second cut are equal. Usually this is avoided by testing and, if necessary, changing the random values before the crossover takes place.

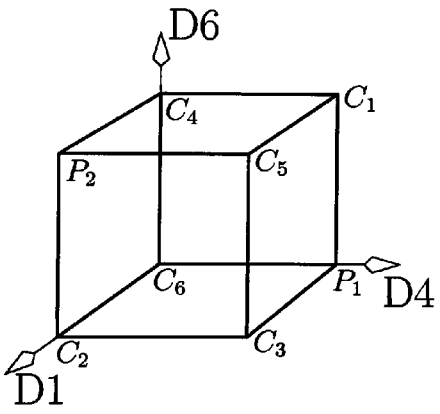


Figure 56: 3-dimensional search-space.

This solution has a drawback: it is only perfect for up to three dimensions. For more dimensions the dual-cut crossover, too, cannot reach every possible child in the hyper-cube in one generation. From the parents $P_1 = \{1, 1, 1, 1\}$ and $P_2 = \{0, 0, 0, 0\}$ the children $C_{13} = \{1, 0, 1, 0\}$ and $C_{14} = \{0, 1, 0, 1\}$ cannot be obtained by dual-cut crossover. A quadruple-cut crossover would be needed. In general of the $2^n - 2$ children only $n(n-1)$ can be reached within one generation. The Tables 14 to 16 list all possible children,

parents	
$P_1 = \{1,1,1,1\}$	$P_2 = \{0,0,0,0\}$
children obtainable by dual-cut crossover	
$C_1 = \{0,1,1,1\}$	$C_2 = \{1,0,0,0\}$
$C_3 = \{1,0,1,1\}$	$C_4 = \{0,1,0,0\}$
$C_5 = \{1,1,0,1\}$	$C_6 = \{0,0,1,0\}$
$C_7 = \{1,1,1,0\}$	$C_8 = \{0,0,0,1\}$
$C_9 = \{0,0,1,1\}$	$C_{10} = \{1,1,0,0\}$
$C_{11} = \{1,0,0,1\}$	$C_{12} = \{0,1,1,0\}$
children not obtainable by dual-cut crossover	
$C_{13} = \{0,1,0,1\}$	$C_{14} = \{1,0,1,0\}$

Table 14: Possible children in the fourth dimension.

parents	
$P_1 = \{1,1,1,1,1\}$	$P_2 = \{0,0,0,0,0\}$
children obtainable by dual-cut crossover	
$C_1 = \{0,1,1,1,1\}$	$C_2 = \{1,0,0,0,0\}$
$C_3 = \{1,0,1,1,1\}$	$C_4 = \{0,1,0,0,0\}$
$C_5 = \{1,1,0,1,1\}$	$C_6 = \{0,0,1,0,0\}$
$C_7 = \{1,1,1,0,1\}$	$C_8 = \{0,0,0,1,0\}$
$C_9 = \{1,1,1,1,0\}$	$C_{10} = \{0,0,0,0,1\}$
$C_{11} = \{0,0,1,1,1\}$	$C_{12} = \{1,1,0,0,0\}$
$C_{13} = \{1,0,0,1,1\}$	$C_{14} = \{0,1,1,0,0\}$
$C_{15} = \{1,1,0,0,1\}$	$C_{16} = \{0,0,1,1,0\}$
$C_{17} = \{1,1,1,0,0\}$	$C_{18} = \{0,0,0,1,1\}$
$C_{19} = \{0,1,1,1,0\}$	$C_{20} = \{1,0,0,0,1\}$
children not obtainable by dual-cut crossover	
$C_{21} = \{0,1,0,1,1\}$	$C_{22} = \{1,0,1,0,0\}$
$C_{23} = \{1,0,1,0,1\}$	$C_{24} = \{0,1,0,1,0\}$
$C_{25} = \{1,1,0,1,0\}$	$C_{26} = \{0,0,1,0,1\}$
$C_{27} = \{0,1,1,0,1\}$	$C_{28} = \{1,0,0,1,0\}$
$C_{29} = \{1,0,1,1,0\}$	$C_{30} = \{0,1,0,0,1\}$

Table 15: Possible children in the fifth dimension.

parents	
$P_1 = \{1,1,1,1,1,1\}$	$P_2 = \{0,0,0,0,0,0\}$
children obtainable by dual-cut crossover	
$C_1 = \{0,1,1,1,1,1\}$	$C_2 = \{1,0,0,0,0,0\}$
$C_3 = \{1,0,1,1,1,1\}$	$C_4 = \{0,1,0,0,0,0\}$
$C_5 = \{1,1,0,1,1,1\}$	$C_6 = \{0,0,1,0,0,0\}$
$C_7 = \{1,1,1,0,1,1\}$	$C_8 = \{0,0,0,1,0,0\}$
$C_9 = \{1,1,1,1,0,1\}$	$C_{10} = \{0,0,0,0,1,0\}$
$C_{11} = \{1,1,1,1,1,0\}$	$C_{12} = \{0,0,0,0,0,1\}$
$C_{13} = \{0,0,1,1,1,1\}$	$C_{14} = \{1,1,0,0,0,0\}$
$C_{15} = \{1,0,0,1,1,1\}$	$C_{16} = \{0,1,1,0,0,0\}$
$C_{17} = \{1,1,0,0,1,1\}$	$C_{18} = \{0,0,1,1,0,0\}$
$C_{19} = \{1,1,1,0,0,1\}$	$C_{20} = \{0,0,0,1,1,0\}$
$C_{21} = \{1,1,1,1,0,0\}$	$C_{22} = \{0,0,0,0,1,1\}$
$C_{23} = \{0,1,1,1,1,0\}$	$C_{24} = \{1,0,0,0,0,1\}$
$C_{25} = \{0,0,0,1,1,1\}$	$C_{26} = \{1,1,1,0,0,0\}$
$C_{27} = \{1,0,0,0,1,1\}$	$C_{28} = \{0,1,1,1,0,0\}$
$C_{29} = \{1,1,0,0,0,1\}$	$C_{30} = \{0,0,1,1,1,0\}$
children not obtainable by dual-cut crossover	
$C_{31} = \{0,1,0,1,1,1\}$	$C_{32} = \{1,0,1,0,0,0\}$
$C_{33} = \{1,0,1,0,1,1\}$	$C_{34} = \{0,1,0,1,0,0\}$
$C_{35} = \{1,1,0,1,0,1\}$	$C_{36} = \{0,0,1,0,1,0\}$
$C_{37} = \{1,1,1,0,1,0\}$	$C_{38} = \{0,0,0,1,0,1\}$
$C_{39} = \{0,1,1,1,0,1\}$	$C_{40} = \{1,0,0,0,1,0\}$
$C_{41} = \{1,0,1,1,1,0\}$	$C_{42} = \{0,1,0,0,0,1\}$
$C_{43} = \{0,1,1,0,1,1\}$	$C_{44} = \{1,0,0,1,0,0\}$
$C_{45} = \{1,0,1,1,0,1\}$	$C_{46} = \{0,1,0,0,1,0\}$
$C_{47} = \{1,1,0,1,1,0\}$	$C_{48} = \{0,0,1,0,0,1\}$
$C_{49} = \{0,0,1,0,1,1\}$	$C_{50} = \{1,1,0,1,0,0\}$
$C_{51} = \{1,0,0,1,0,1\}$	$C_{52} = \{0,1,1,0,1,0\}$
$C_{53} = \{1,1,0,0,1,0\}$	$C_{54} = \{0,0,1,1,0,1\}$
$C_{55} = \{0,1,1,0,0,1\}$	$C_{56} = \{1,0,0,1,1,0\}$
$C_{57} = \{1,0,1,1,0,0\}$	$C_{58} = \{0,1,0,0,1,1\}$
$C_{59} = \{0,1,0,1,1,0\}$	$C_{60} = \{1,0,1,0,0,1\}$
$C_{61} = \{0,1,0,1,0,1\}$	$C_{62} = \{1,0,1,0,1,0\}$

Table 16: Possible children in the sixth dimension.

obtainable and not obtainable. This speed-up problem is one reason for the introduction of the shuffle-cut crossover.

2.8.2.3 Shuffle Cut

Another problem with the above mentioned crossover methods is that neighbouring genes will almost always stay together, especially if the DNA-string is very long as in the case of the application described in this book.

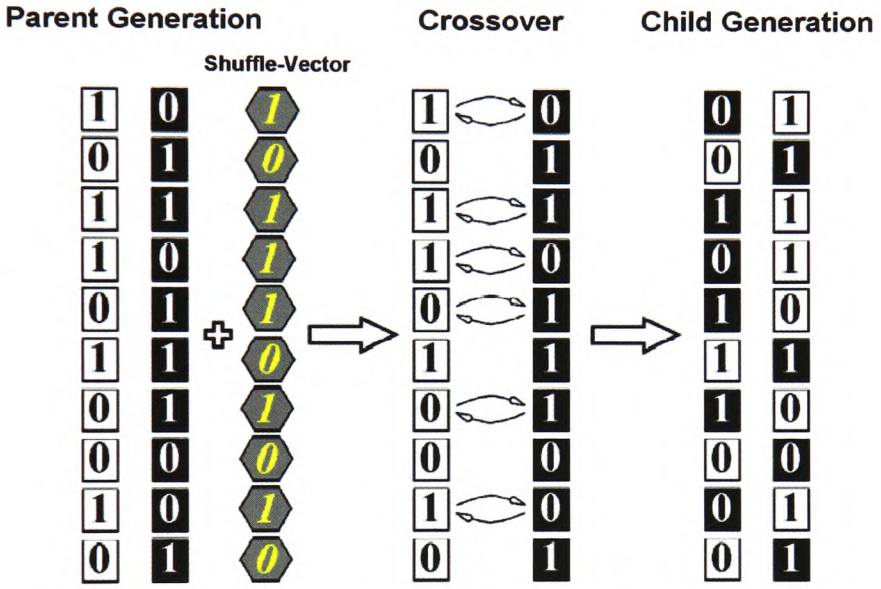


Figure 57: Shuffle-cut crossover.

A way of avoiding this is by using the shuffle crossover method. A special shuffle-DNA-string of the same size as the other DNA-strings is created randomly for every new generation. The genes of the parent DNA-strings are exchanged if the according gene of the shuffle-DNA-string is set to one, not exchanged if set to zero. Fig. 57 and Eqs. 58 and 59 will clarify this procedure.

$$\left.\begin{array}{rcl} C_1(i) & = & P_1(i) \\ C_2(i) & = & P_2(i) \end{array}\right\} \quad \forall \quad R(i) < P_c \tag{58}$$

$$\left.\begin{array}{rcl} C_1(i) & = & P_2(i) \\ C_2(i) & = & P_1(i) \end{array}\right\} \quad \forall \quad R(i) \geq P_c \tag{59}$$

With $R(i) \in [0, 1]$ being the random value for the i-th DNA-string element, thresholded by a user-defined crossover probability value P_c. For $P_c = 0.5$ the shuffle-cut crossover is unbiased[25], biased otherwise.

The shuffle-cut crossover is also known as uniform crossover, and the shuffle-DNA-string as template mask.

2.8.3 Mutation

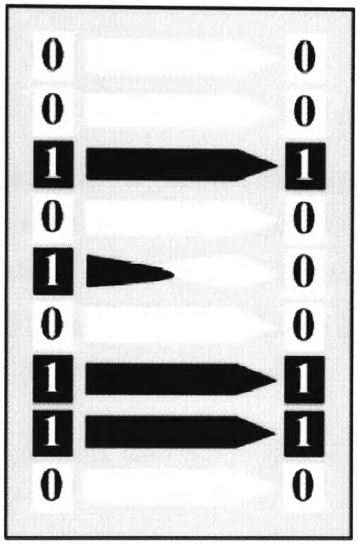


Figure 58: Mutation.

Mutation is needed to avoid finding only local optimal solution. This can happen if throughout the population a certain gene is set to one, or zero accordingly, like the third, sixth or eighth gene in the figure 53. By means of crossover the influence of this gene will never again change, it is called a 'fixed gene'. The possible solutions that can be addressed are halved with every fixed gene. Mutation now toggles the influence of a randomly chosen gene from one to zero and vice versa and thus increases the search space if a fixed gene is mutated. (See Fig. 58)

It is important to note that the mutation algorithm does not search for fixed genes, but mutates genes randomly.

[25]The single-cut and dual-cut crossover variants are unbiased by definition. A biased version, a sensible option only for the dual-cut crossover, would need a non-linear transformation of the random values.

2.8.4 Dimwits

Dimwits are DNA-strings with an under par fitness value. Alternatively the least capable may be chosen. One or more dimwits are added to the group of parents and being used for the crossover process. Again the reason is to surpass local optima. A dimwit might has an optimal gene sequence not found in the group of the fittest parents.

Tests have shown that mutation and adding dimwits are important for very long DNA-strings. This is due to the fact that in such cases the population size is for computational reasons usually significantly smaller than the number of genes in a DNA-string.

2.8.5 Genealogy

A number of different mating procedures can be adopted. The *brute force* method is to mate every individual of the population to all the others. A parent population size of m will produce $m^2 - m$ children, all of which have to be rated for their individual fitness.

A way of producing less children—and thus fewer fitness tests—is mating by rank. Possibilities that produce m children are for example

$$\{[1 \circledcirc 2], [3 \circledcirc 4], [5 \circledcirc 6], \cdots, [m-1 \circledcirc m]\} \tag{60}$$

called conalliance, and

$$\{[1 \circledcirc m], [2 \circledcirc m-1], [3 \circledcirc m-2], \cdots, [m/2-1 \circledcirc m/2]\} \tag{61}$$

called mésalliance.

A variant common in the animal kingdom is to mate the fittest, the α-individual, to all others, producing $2(m-1)$ children:

$$\{[1 \circledcirc 2], [1 \circledcirc 3], [1 \circledcirc 4], \cdots, [1 \circledcirc m]\} \tag{62}$$

A further variant is to use the two, three or p fittest individuals for mating with all others:

$$\{[i \circledcirc j]\} \quad \forall \quad i \in \{1, p\}, j \in \{i+1, m\}, p < m \tag{63}$$

Here the census counts $p(2m-p-1)$ children. For $p = m-1$ this method is identical to the *brute force* method mentioned above.

2.8.6 Fitness Function and Selection

Next to the crossover process itself the fitness evaluation is most important. Fitness evaluation is the performance test of the system using every newly created DNA of the population—the older ones do not have to be tested again—and thus a number of sets of selected features. After those tests only the better DNA-strings—those that yield a better performance of the system which is being optimized—stay in the population and the production of a new population starts again. To keep the population from growing indefinitely only the m fittest individuals are kept. At that point there is no distinction between parents and children.

2.8.7 Cut Off

The mating and testing process continues as long as the fitness differences between parent and child population are significant for a specified number of generations. Test have shown that very often all the children of one generation have a lower fitness than their parents. Therefore it is necessary to produce and test more generations before an individual is declared the fittest member of the population.

To speed up the fitness testing the new DNA-strings are checked for equal parents or siblings. Tests have shown that even with a very quick fitness test the speed up is significant.

Building and testing a complete genealogy-tree will be useful in the case of a longer fitness test. All new children are then tested for identical ancestors before the fitness test is being performed. Every member of the population that ever existed during the application of the algorithm will be kept as an entry in a binary tree. Thus it is possible to quickly search for identical entries. This part has yet to be included into the system.

In the case of this Texture Classification System the fitness describes the ability to distinguish between different textures. Fig. 46 gives an idea how the features that describe a certain texture build a cluster in the feature plane. The aim is to find such a selection of features for which the clusters do not intersect with one another. The better the clusters are kept apart, the easier the classification is.

A description of how the GA is used in the Texture Classification System is given in Section 3.4. More information on GAs can be found in [Gol89, Hol92a, Hol92b, Jon85, Kar91, Küh95, Lei95, Neu96b, Neu96c, Neu97, PKL94, SHF94, Wal95, Wig92].

2.9 Summary

The first part of this chapter gave a detailed introduction into the complexity of natural vision with a strong emphasis on the human visual system. Even though a lot of questions concerning the human vision and especially the brain have yet to be answered by scientific research it became clear that an artificial vision system has to use a combination of many image processing and classification techniques to be able to provide solutions for the task of texture classification.

The chapter's second part provided a detailed introduction into textures. The differences between artificial and genuine, as well as regular and statistical texture have been explained. Additionally a new classification into subgroups has been introduced. Also the specific characteristics of multi-textural images have been addressed.

The information described in the first two sections of this chapter will be used for the construction of the main frame of the Texture Classification System.

The following two sections of this chapter saw the introduction of numerous methods that can be used for texture identification. It became clear that the necessary features of the texture images can be best extracted by using a combination of image filtering and transformation and statistical analysis of the such reduced image contents. Some new approaches have been described and their possible use discussed.

Different methods can be used for the classification of the feature vector. Taking into account the possible size of the feature vector the necessity to introduce a system optimisation becomes obvious. The potential methods for classification and optimisation have been discussed in detail.

It became clear that neither a single method, nor some sequentially or parallel used methods will provide the features needed for a substantial classification. The next chapters show how the Texture Classification System solves this problem.

Chapter 3

The Texture Classification System

3.1 Objectives

This is the part of the project were the author's new and unique approach for the design and construction of the Texture Classification System is being explained. The prior chapters already showed that a complex design is necessary to achieve useful results. As this has not been done before the ideas and structure of this new system design are being explained in detail.

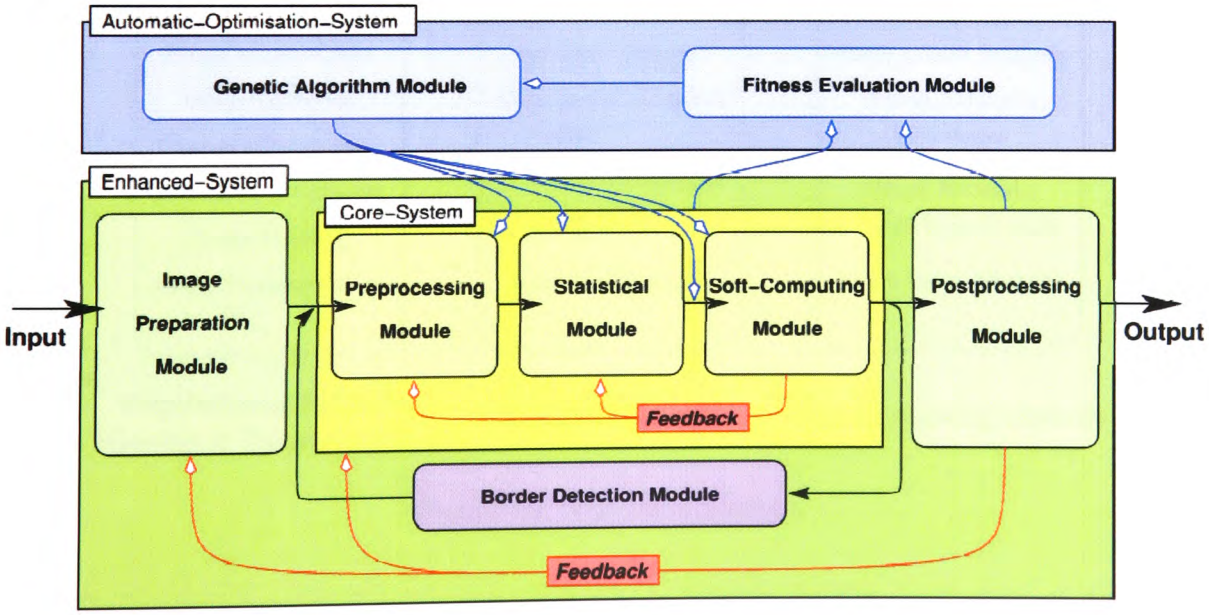


Figure 59: The Texture Classification System.

The objective is to construct a system that can handle the data flow through the modules described in Chapter 2. The modules have to be easily interchangeable to allow for better algorithms or different approaches in future releases. Setting parameters has to be easy and it has to be possible to automatically adjust parameters from within the system. Therefore it is necessary to allow information feedback and thus automatic control loops.

As the input images will be multi-textural images this, too, has to be handled. Sub-images have to be extracted and from the result borders between textures have to be found.

An optimisation of the system shall be possible. Due to the complexity only an automatic optimisation will be possible. Such a module has to be added to the system.

3.2 The Core-System

3.2.1 General Introduction

The Core-System (CS) can be divided into three major parts plus input and output facilities.

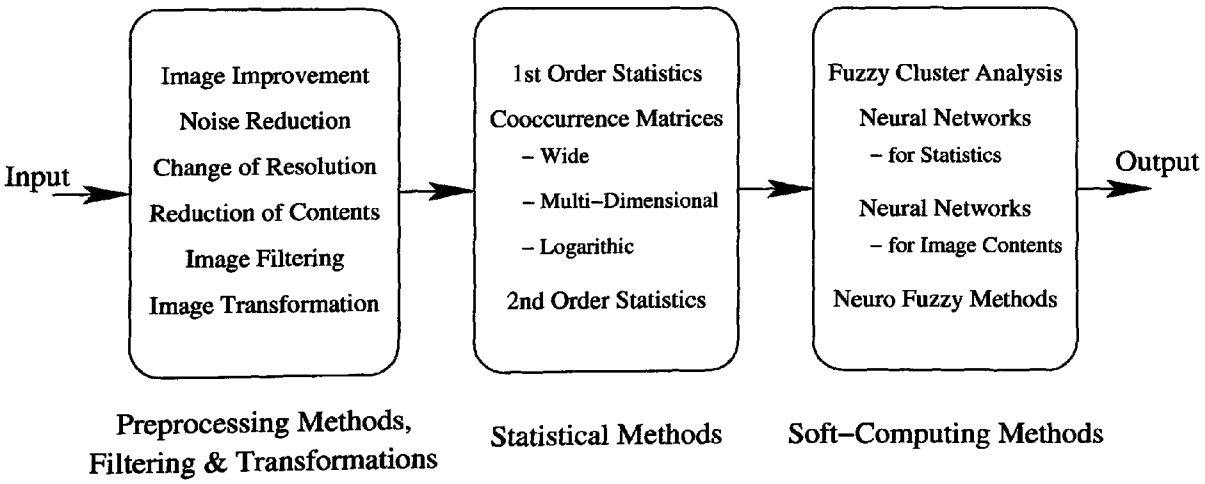


Figure 60: The Core-System.

Those parts consist of preprocessing methods followed by the application of statistical methods and thirdly a variety of soft-computing methods.

Even though the figure 60 suggests a one-shot solution, this is not the case for the Core-System (CS). At several levels of the whole system feedback facilities are included which are used to adjust parameters according to the β-output[1] (cf. Section 5.2).

Such a modularised and parallelised system with additional feedback facilities has not been designed before and is one of the major novelties of the system.

3.2.2 Input

The input for the CS is a single textured subimage, which again is the output of the first module of the Enhanced-System (ES). Depending on the size and resolution of the texture[2] the parameters of some of the preprocessing methods are adjusted.

For each textured subimage at the input a classification result is expected at the output, which in turn is the input for the third part of the Enhanced-System (ES).

3.2.3 Preprocessing Methods

For the subimage—in the following referred to as image or texture—improvement might be necessary, e. g. histogram shift or noise reduction. Depending on the image and the proposed transformation and statistical methods a change of resolution can be of advantage. Another preprocessing method is the reduction of the image contents, i. e. using smoothing filters to loose redundant detail information. The last two methods can be combined with edge detection filters, gradient operators or other transforming methods described in Section 2.4. Thereby a great variety of altered images are produced all of which provide features detectable by the statistical methods.

These are methods used in singular form in contemporary image processing systems (cf. [Abm94, BB82, BB93, Ern91, Hab91, Ric93, SHB93, Ste93b]). The Texture Classification Systemuses not one specialised combination of methods, but in the first instance all available methods in a multitude of combinations.[3]

[1] The β-output of a module is used to change the parameters of preceding modules until the output is satisfactory. If no rulebase or look-up-table is available for this task GAs could be utilised.

[2] A texture is often described as an amalgamation of so called texels, single texture elements. This is only in so far a usable characterisation as man-made textures are being described. The terms 'size' and 'resolution' refer to the size of a (possible) texel in comparison to the texture subimage and to its resolution in pixels per texel.

[3] The AOS described in Section 3.4 will reduce the number of methods to make a lean system.

3.2.4 Statistical Methods

The statistical methods used in the Texture Classification System have been described in detail in Section 2.5. Applying the methods available for the system, i. e. the mean, variance, skewness, kurtosis, and entropy of the 1st order statistics (cf. Section 2.5.1) and the energy, contrast, entropy, and correlation for all matrices of the WCM (cf. Section 2.5.2.2), will provide a feature vector of 101 elements. If only one of the LCMs described by the Eqs. 42 to 45 in Section 2.5.2.5 is used in combination with the WCMs, additional 96 feature values are being obtained.

These are the features for only one filtered and/or transformed image, of which there are many[4]. Fig. 61 shows the principle idea.

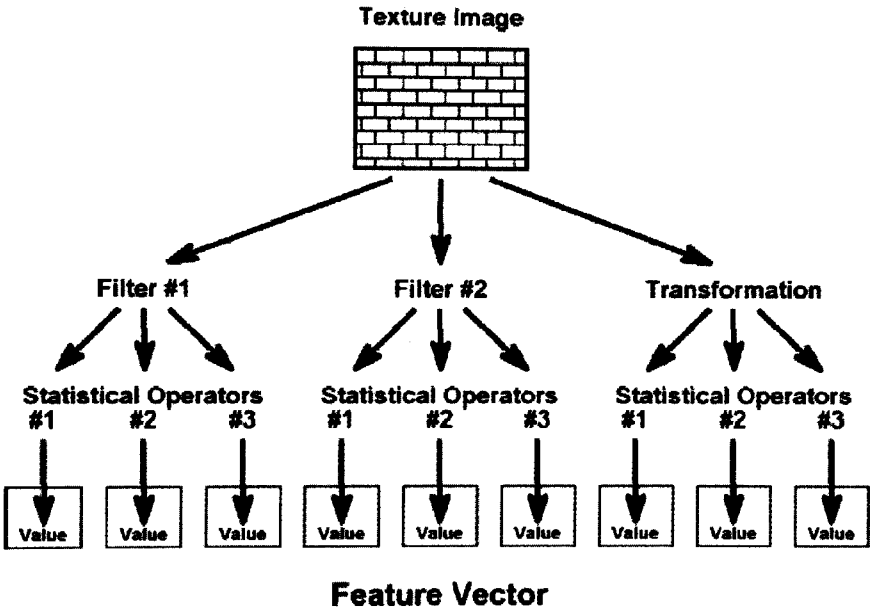


Figure 61: Principle assembly of a feature vector.

3.2.5 Soft-Computing Methods

This is the most important part of the system, and the most versatile one. The Soft-Computing (SC) methods are being used to extract the information from the feature vectors obtained in earlier stages.

[4]How many there are will be described in Section 5.2

Different SC[5] methods can be used in this part of the system. Thus it can be estimated which method or which combination of methods are best suited for the task. The estimation can be done by user experience, but this is only the second best solution as the user's knowledge is usually limited to specialised areas and thus often biased. An unbiased selection of methods can be achieved by using the Automatic-Optimisation-System, described in Section 3.4.

3.2.5.1 Fuzzy Cluster Analysis

Using sharp or fuzzy clustering for the classification of the feature vectors is a computationally not too intensive approach. Even though the dimension of the feature vector space can be very high (cf. Section 5.2) clustering can provide results even when only few test cases are available.

The fuzzy approach is useful if the sharp clustering is not classifying correctly, or, and this is important in respect to the available computational resources, to further reduce the number of features used for classification. Additionally the Fuzzy Cluster Analysis (FCA) gives information on the membership value for each cluster. As a cluster represents a distinct texture, this information is useful to handle new or disturbed textures.

Section 2.6 provides more information on clustering and fuzzy clustering.

3.2.5.2 Neural Network

An alternative to the clustering approach is the employment of neural networks. As has been described in 2.7 many different NN designs exist. It is important to choose the type which is most appropriate for the task. Again the choice is best performed automatically by using the Automatic-Optimisation-System.

Even though some of the NNs seem to be very promising, they are not the prominent choice. This is mainly due to the limited computationally resources available. Even when using the latest processor type the computers are not in the position to handle networks with a couple of million neurons. Therefore networks have to be used that are not too demanding when operated.

[5]Soft-Computing (SC) is a generic term for all modern software technologies like Fuzzy Logic (FL), Neural Networks (NNs), Genetic Algorithms (GAs), Evolutionary Programming (EP) and Chaos Theory (CT) to name the major branches. The combination of the SC methods among one another and with traditional methods is becoming more and more important for solving complex problems, especially AI-complete problems.

To cut down on development time next to some own implementations the Stuttgarter Neuronale Netze Simulator (SNNS) is used. More information on that software can be found in Section D.1.

3.2.5.3 Neuro Fuzzy and Fuzzy Neural Networks

Neuro Fuzzy methods and Fuzzy Neural Networks are new fields in SC where two established methodologies are combined. There are different ways of getting Fuzzy Logic (FL) and NNs together. The easiest way is a hybrid system, where a fuzzy module interchanges data with a neural module. This is the case in the CS provided both modules are enabled at the same time. More complex combinations are Neuro Fuzzy (NF) systems and FNNs. So far no modules are implemented for the Texture Classification System. General articles on NF systems and FNNs are in [Cox92, Feu94, FL95a, FL95b, Feu95, HG94, Hun93, KS93, Kos92, LFM95, LFT94, NKK93, NKK94, NK95, NNK96, Neu96a, Zad93b, ZD93].

3.2.6 Output

The output of the CS is plain text which references a specific texture. As single texture images are used only an identification has to be made. In the case of the Fuzzy Clustering System (FCS) being used this identification is preceded by a membership value for each cluster of the plane. This information, fuzzy as it still is, can be defuzzified in a postprocessing system. That is part of the ES.

3.3 The Enhanced-System

After the Core-System (CS) has been successfully designed as described in Section 3.2 those modules have to be included that enable the system to operate in a real world environment. Therefore the Enhanced-System (ES) has been developed together with a variety of modules for pre- and postprocessing.

3.3.1 General Introduction

The ES can be divided into three major parts plus input and output facilities.

The attentive reader will have guessed that the ES—just like the CS—incorporates multiple feedback facilities. See Section 5.2 for detailed information.

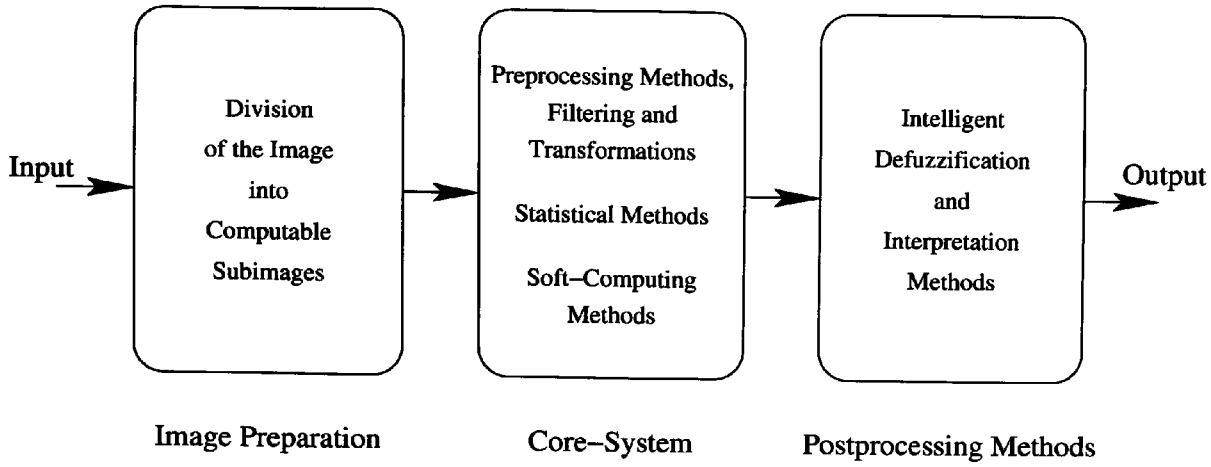


Figure 62: The Enhanced-System.

3.3.2 Input

In contrast to the CS the input images are multi-texture images, e. g. x-ray images, satellite or aerial photography, etc.

The size of the image must be big enough to obtain an appropriate texture resolution to image resolution factor. If the single texture areas within the image are too small, a texture identification will not be possible, as is described in detail in Section 3.5. A subimage must at least contain one full texel. As texels are usually not identical or—in natural textures—not even clearly definable a greater number should be used. The subimage itself should neither be too small, as otherwise the texture resolution would suffer if above mentioned conditions are obeyed, nor too big for the sake of wasted computational resources.

3.3.3 Image Preparation

The multi-texture image has to be divided into small subimages. All subimages, of which the majority should only contain one type of texture, have to be computed sequentially.[6] The CS is used for that task. Feedback from the CS or the postprocessing modules of the ES can be used to control the image preparation.

[6]It of course is possible to perform the computation in parallel on a specialised hardware platform or even under a multi-tasking operating system, but the result will be in no way different to a sequential computation.

Difficult and highly interesting areas are those that contain more than one type of texture. In this case the output of the CS is not easily predictable as the feature vectors might give information that will make believe, that a certain, well defined texture has been computed. Therefore it is necessary to compute overlapping subimages once a change in the type of texture has occurred. That means that the image preparation depends on the output of the CS. How this subimage overlapping technique operates is described in Section 3.5.

3.3.4 Core-System

The CS has been described in Section 3.2. It uses preprocessing, statistical and soft-computing methods to identify single textures. Membership values are given at the output if fuzzy clustering or certain types of NNs are used for classification. That means, that the CS will not necessarily give a clear identification. Therefore some intelligent postprocessing methods are employed.

3.3.5 Postprocessing Methods

This final part of the whole system uses some intelligent defuzzification and interpretation methods. Those methods are highly context dependent. Expert knowledge of the image contents is important for the classification of image objects and detection of boundaries between objects.

A straight forward boundary detection and area segmentation technique is described in Section 3.5. Other intelligent methods have to be developed to step in where the straight forward method fails.

The measure for an adequate postprocessing method is the human detection and classification capabilities. The goal is to equal or even better the human perceptual abilities.

3.3.6 Output

The output depends on the role the Texture Classification System plays for solving the problem of the particular image analysis task. In general the output will be a representation of the original image in which texture areas are identified and classified and

thus boundaries between textures drawn. The whole input image is now described by its texture properties and the location of specific textures.

3.4 The Automatic Optimisation

3.4.1 Feature Vector Optimisation

As the number of features generated in the previous modules of the system is very large—easily exceeding 10^6 as shown in Section 5.2 and Figure 107—it is necessary to select relevant features for the classification by either clustering or neural networks. Doing this manually is not an option[7] as the dimension of the feature plane is by far too large to be visualisable and the possible interconnections between features too complex. Therefore an automatic feature selector has to be included into the system.

3.4.1.1 Feature Selection using Genetic Algorithms

An ideal as fully automatic way to reduce the size of the feature vectors is to apply a Genetic Algorithm (GA). For the system test a starting population of size p is created where each member has as many genes as there are features. If for a certain member M_k a certain gene G_i is randomly set to zero, the according feature F_i will not be regarded in the following modules of the system, i. e. the classification modules. This does not mean that this feature F_i is ruled out from being used in the Texture Classification System. Other members of the population might have G_i set to one. Employing a population that uses only subsets of the original feature vectors for each member of the population splits the so far linear processing of the system into p branches. Each branch is then tested for its fitness, which is in this case the classification quality.

After the starting generation of the population has been tested and ranked by its fitness, a new generation of members for the population is produces. This is done by the mating procedures described in detail in Section 2.8.

As the gene values of the starting population are set by random, the number of features in each branch is statistically half the number of original features. For a very large feature vectors as in the case of the Texture Classification System these are still

[7]Even if many of the modules that produce texture features are excluded randomly, the feature vector is still too large.

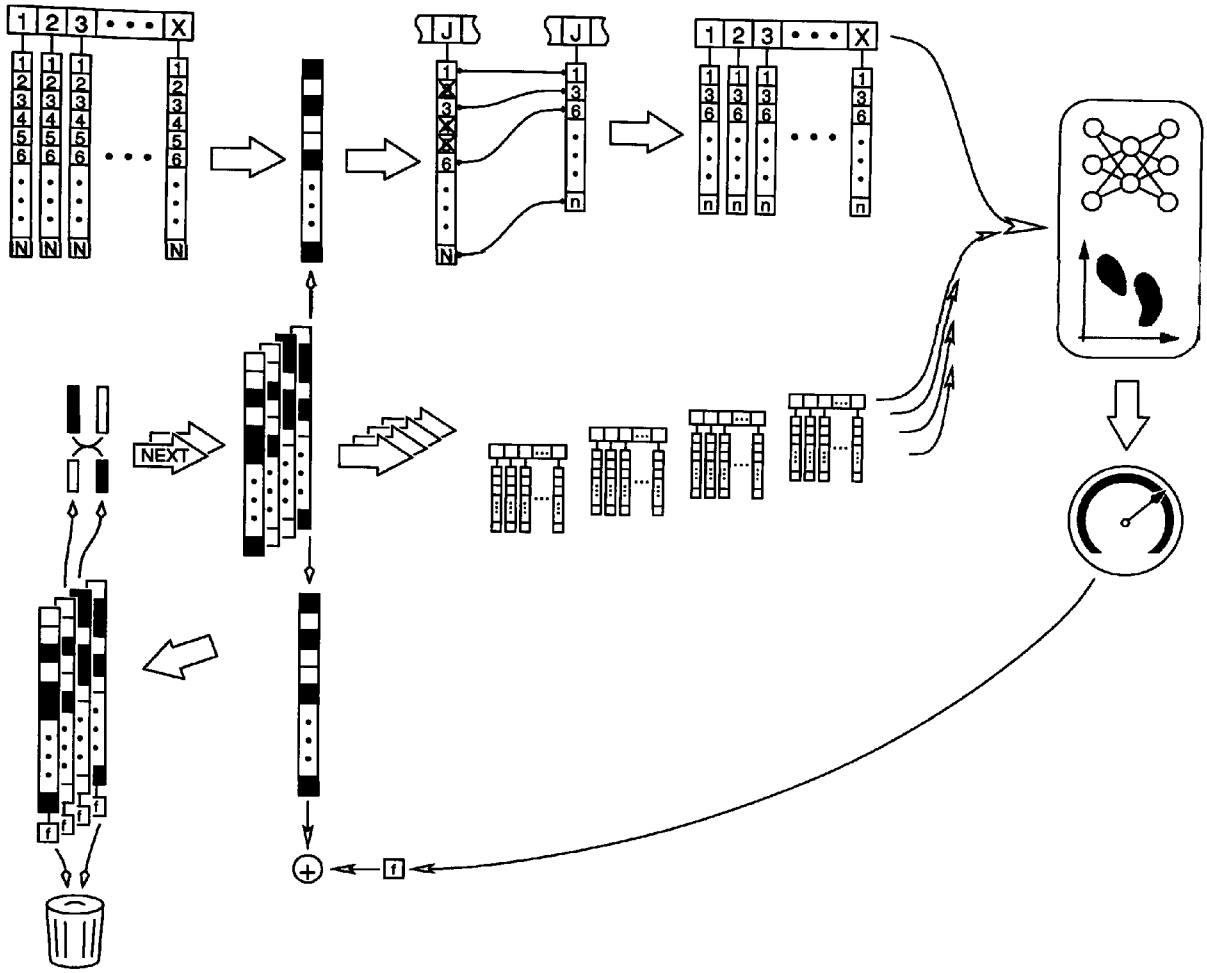
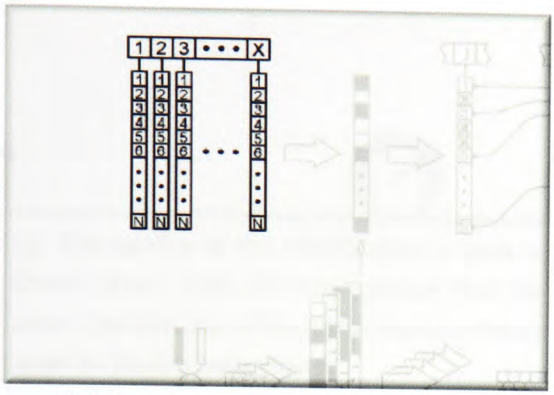


Figure 63: Sketch of the feature selection and fitness test. A detailed explanation can be found in Fig. 64(a)–64(j).

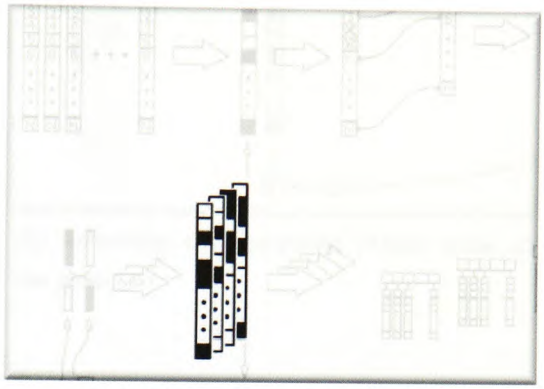
very many features that have to be used for classification. This might not be possible with regard to computational resources. Therefore the feature vector subsets are again being reduced in size by the application of another genetic algorithm. By this way, after some iterations, the feature vector is reduced to a size that can be computed. This of course means that a large number of branches now exist that have to be tested for their individual fitness. Additionally that means that always only few features are being used at a time. The sketch in Fig. 63 shows the principle.

To speed up the system the first iteration of feature vector optimisation stops at a fitness rate of e. g. 80%. The best or m best members of the population are then tested for genes fixed to zero throughout the population. Those 'dead genes' are excluded from

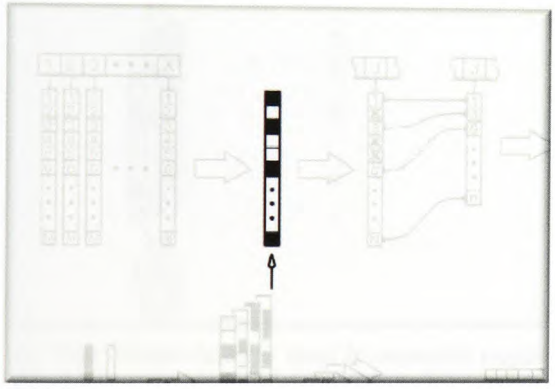
the feature vector permanently. This reduces the search space as these genes cannot be reanimated by mutation. Thereafter the optimisation procedure continues for the reduced feature vector.



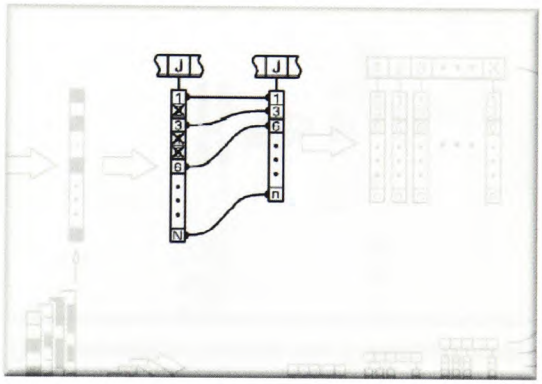
(a) The input vector of feature vectors.



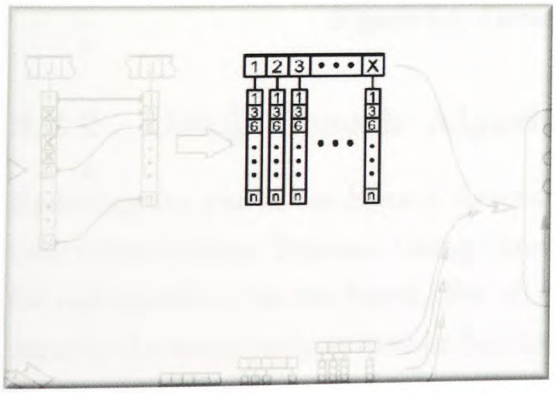
(b) The (starting) population.



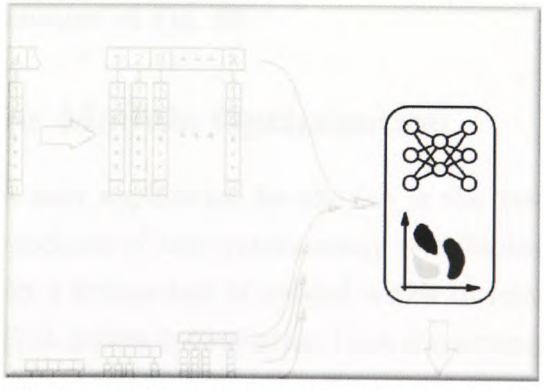
(c) One gene at a time is used as feature selector, until all genes of the population have been used.



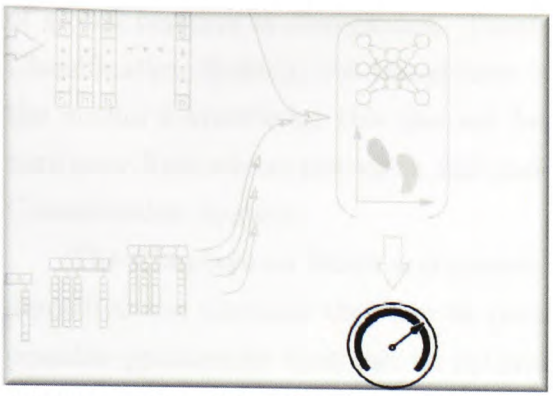
(d) Reduction of the number of features. All the feature vectors of the input vector are being processed in this way.



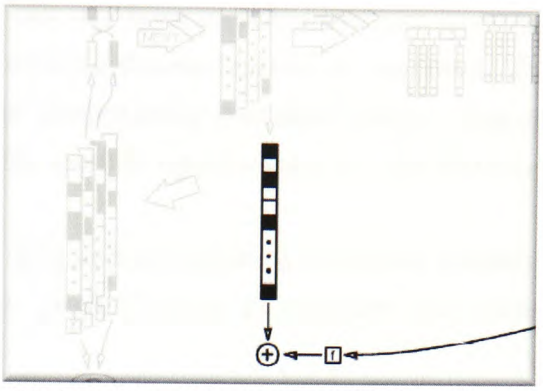
(e) The vector of feature vectors with reduced number of features.



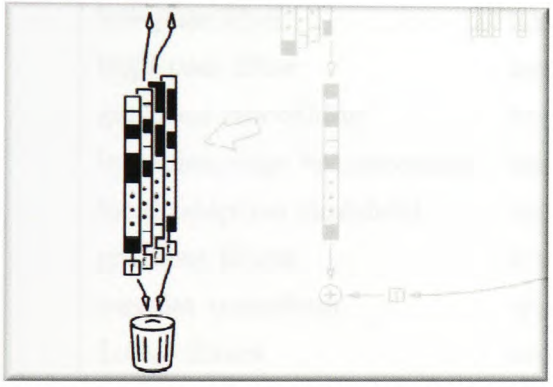
(f) A classification module is used to classify the reduced size feature vectors.



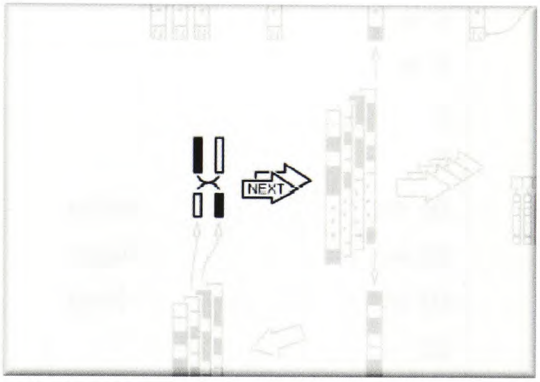
(g) The quality of the classification is used as fitness value. This, of course, means that the exact classification of the tested feature vectors must be known beforehand.



(h) Attaching the computed fitness value to the gene.



(i) The fitness value is used to separate population into good and bad members.



(j) Crossover to produce new population members. After this step the procedure starts all over again with Figs. 64(a) and 64(b) using now the new population members.

Figure 64: Detailed explanation of Fig. 63

3.4.2 Using Genetic Algorithms for Module Optimisation

Reducing the size of the feature vectors is not the only application for the GA in the Texture Classification System. Going through the modules of the system many possibilities for optimisation can be found. For all those tasks a fitness test is needed which in most cases is the same as described in Section 3.4.1. This makes optimisation time consuming, or, for large systems, near impossible due to limited resources. Therefore local fitness tests have to be employed. This is only the second best solution as the interconnectivity

of global features is disregarded. These test have not yet been included into the Texture Classification System, but integration is easy provided a fitness function is available. To the author's knowledge this has not been done or theoretically described before. Again hardware limitations prevent a full scale test of the module optimisation for the Texture Classification System.

The Chapters on filters and transformations (2.4) and on statistics (2.5) have already described the modules that can be parameterised. Table 17 gives an overview over some possible parameters that can be optimised.

Next to setting the parameters the GA is used to decide over taking or not taking a method altogether.

method	1st parameter	2nd parameter	# of set-ups
low-pass filter	kernel size		≈ 5
high-pass filter	kernel size		≈ 5
gaussian smoothing	kernel size		4
laplacian edge enhancement	kernel type		3
local adaptive threshold	method	value	≈ 40
gradient filters	type	result	≈ 10
wavelet transform	type	level	≈ 50
LAWS filters	kernel		25
LAWS TEM	kernels		15
1st order statistics	moments		5
2nd order statistics	neighbour	value	92

Table 17: Some modules with adjustable parameters.

Up to this point the GA changed the contents and size of the feature vector. The next application for the GA is to choose the best classification module, i. e. taking clustering or neural networks, which cluster algorithm or which NN is used, how the NN is designed, etc. The Chapters on clustering (2.6) and neural networks (2.7) introduced the variety of methods.

Last but not least the optimisation module can be used to optimise the optimisation of the above mentioned applications. This means that the set-up of the GA, i. e. crossover method, mating procedure, mutation rate, number of dimwits, etc., and the choice of the fitness function can be optimised, too.

This shows that automatic optimisation methods can be included in almost every module of the system.

3.4.3 The Fitness Function

The fitness function in the Automatic-Optimisation-System (AOS) gives the percentage of the correctly classified test textures as a fitness value.

If this is not good enough for a certain application, a non-linear fitness function could be used. This was not necessary for the Texture Classification System so far, but changing the fitness function is easy. The function has to be monotonous and biased to the range $[0, 1]$. For example a computationally intensive feature will contribute a smaller amount to the fitness value, or the fitness is inversely proportional to the number of features, thus promoting smaller feature vectors.

3.5 The Border Detection Routine

The border detection routine is one of the author's new ideas. The general description in the next section shows how sharp borders in multi-textured images can be identified. In Section 3.5.2 a description is given, how the method can be advanced to detect fuzzy borders. This again is a new idea of the author how to solve one of the problems in multi-texture image segmentation.

3.5.1 General Description

Identifying boundaries between texture areas is not a straight forward task. The boundaries will, provided the texture areas have sharp edges, be pixel sized.[8] But for the texture discrimination detection windows have to be used that are several pixels wide. Therefore the borderline can only be extracted by using overlapping windows. Additionally some confined areas will not be accessible at all (cf. Figs. 70(a) and 70(b)).

The examples in Figs. 65 to 70 show how a border detection is performed. For convenience sharp edged areas are being used and textures are represented by selected colours. The size of the initial image is 240 by 192 pixels and the detection window used is 24 by 24 pixels wide.

[8] For fuzzy edged texture areas cf. page 107.

Figure 65: Image with sharp borders.

At first the whole image is tiled into subimages. The subimages are identical in size to the detection window.

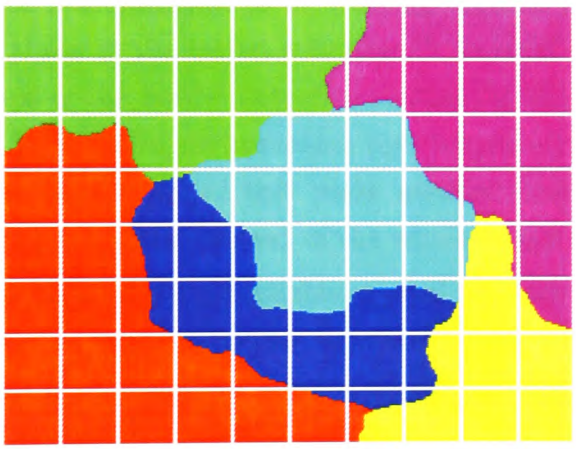
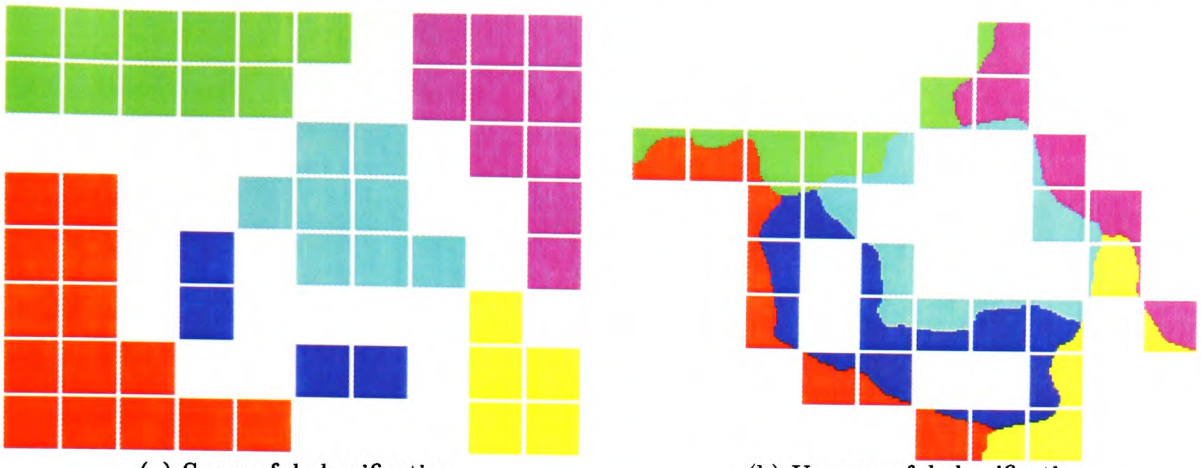


Figure 66: Image tiled into subimages.

Each subimage is processed through the CS. The feature vector of each subimage is classified by the classification routine, e. g. used as the input vector for a neural network or is matched to the appropriate cluster of a FCS, and an output image is being produced. In this output image the identified areas are marked (cf. Fig. 67(a)) as well as those subimages for which no representation within the classification module exists (cf. Fig. 67(b)).[9]

[9]For better visualisation the result image uses the actual subimages instead of a mere alphanumerical representation.



(a) Successful classification. (b) Unsuccessful classification.

Figure 67: Successfully and unsuccessfully classified subimages.

At this stage it must be mentioned that the feature vector of a multi-texture subimage could be identical or near identical to an existing texture, one that might even be completely different to those in the tested subimage.[10] In most cases this will be rectified within the next steps of the border detection routine. If this mismatching proves to be a hazard—that could happen in environments that handle very many textures—some intelligent methods have to be applied that make guesses about the possibility of the occurrence of certain textures in the context of the whole image.

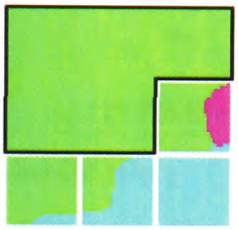
The next step is to check the eight-neighbourhood of every identified subimage. If each neighbour is of the same type, no further steps have to be taken. There are obviously no borders in that area. If on the other hand a neighbouring subimage is dissimilar, a border has occurred.

To home in on the border following steps of the feedback circuit have to be performed. At first an overlapping eight-neighbourhood is used. Each neighbour is therefore moved by half its width or height towards the centre of the subimage in question.[11] Thereafter the newly formed subimages are tested for their contents and their similarity to the centre subimage. Figs. 68(a) to 68(d) are given as an example.

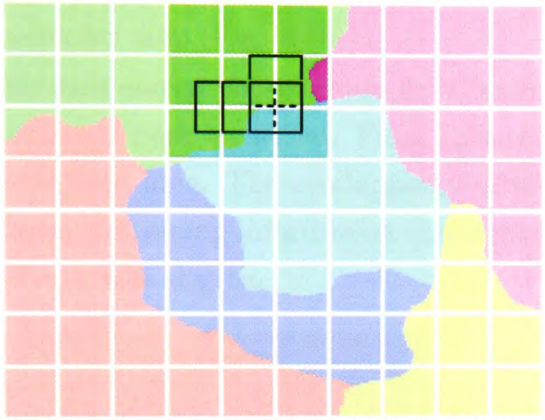
[10]If a texture is 'identified' that is different to the textures of all surrounding subimages it could be corrected with the help of intelligent postprocessing methods. If on the other hand the multi-texture subimage in question has similar feature vectors to one of the neighbouring subimages a postprocessing correction is not possible.

[11]This task has only to be performed by those neighbouring subimages that are dissimilar. For all others this is obsolete as the result is already known.

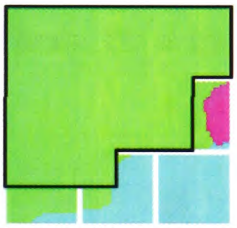
(a) Eight-neighbourhood of a subimage.



(b) Identification result.



(c) Closed in eight-neighbourhood of a subimage.



(d) Identification result.

Figure 68: Eight-neighbourhood and closed in eight-neighbourhood of a subimage and their identification results.

If similarity is found the identified texture area can be increased by the non-overlapping part of the overlapping neighbour. Fig. 69(a) shows the result for the whole image.

Thereafter the neighbouring subimage will move away from the central subimage by a quarter of its width or height. If a dissimilarity occurred at the last step it will move towards the central subimage by a quarter. Again the new subimages are tested and in a next step are moved nearer to the border. Fig. 69(b) shows the result of the quarter or three-quarter overlap respectively for the whole image.

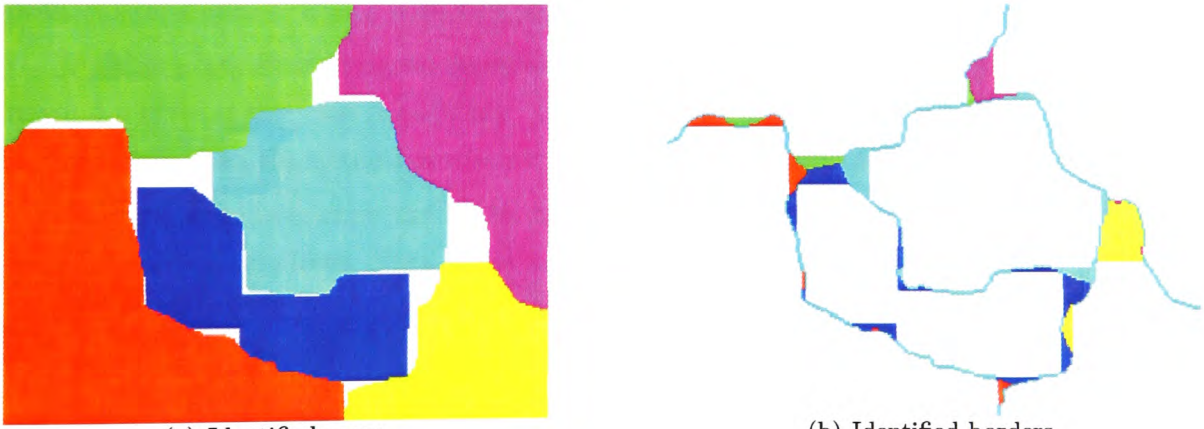
This feedback loop will continue until the shift of the subimages is down to one pixel. Thus every area that can be fully covered by a detection window has been clearly identified. Fig. 69(c) shows the result of the pixel overlap for the whole image.

(a) Half-overlapping. (b) Quarter-overlapping. (c) Pixel-overlapping.

Figure 69: Result of overlapping subimage classification.

As a result a representation of the texture areas (Fig. 70(a)) is obtained or, which is similar, a border map (Fig. 70(b)).

In this deliberately chosen example it can be clearly observed that some areas near to borders have not been identified. These areas are smaller than the detection window size and in a confined area. The overlapping technique does not work under such conditions by definition. Obtaining information about such areas is difficult. SC technology is able to help to a certain degree.



(a) Identified areas. (b) Identified borders.

Figure 70: Result images of the border detection routine.

3.5.2 Fuzzy Border Detection

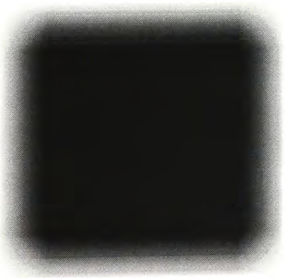
The same problem of having not clearly identifiable areas occurs when images with fuzzy borders have to be handled. Fig. 71 shows the image of Fig. 65 but with unsharp boundaries between the texture areas.

Figure 71: Image with fuzzy borders.

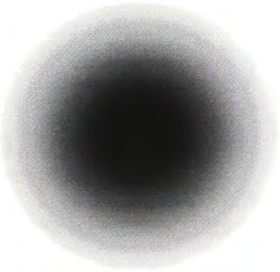
Again a clear identification cannot be obtained for a fairly wide stretch around the borderline. But luckily a clear identification is not necessary if a FCS is used for discrimination. The feature vector for besmeared subimages will move away from the centre of the appropriate cluster but a classification is still possible. A borderline is then defined between two neighbouring textures when the feature vectors have an equal distance to both cluster-centres. Using NNs will provide similar results.

At this point it must again be stressed that the detection window cannot be reduced beyond a certain size. An (imaginary) texel of every texture occurring in the image must not be larger than the detection window. Otherwise a clear identification cannot be made.

An option to overcome the above mentioned restrictions is to use centre weighted or round detection windows. Figs. 72(a) and 72(b) show two possible examples.



(a) Centre weighted square detection window



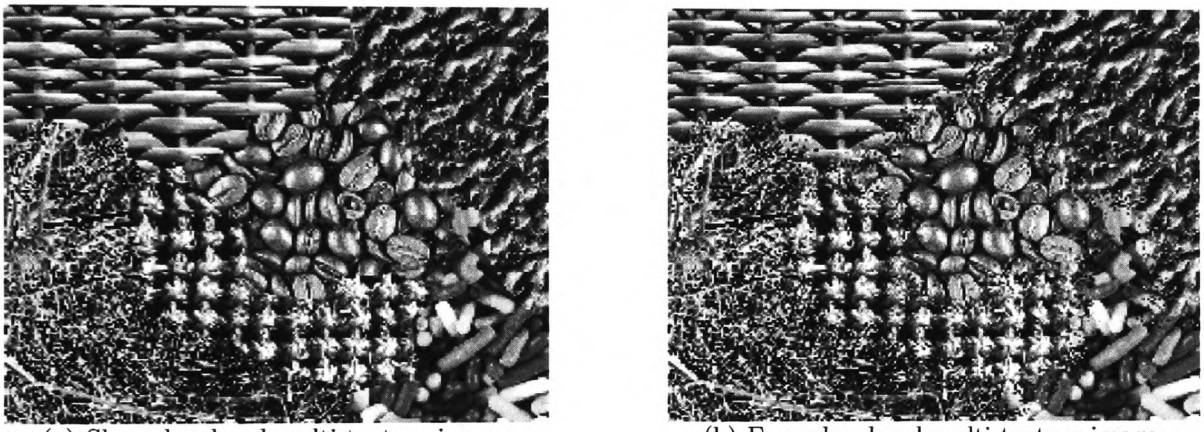
(b) Centre weighted circular detection window

Figure 72: A couple of possible centre weighted detection windows.

Further options are to use ideas developed for segmenting merged characters [Bay89]. Such ideas have not yet been tested for texture classification.

It should be mentioned that ordinary edge detection techniques as found in literature cannot be used. The textures may use all possible greylevels at a time. That means that any gradients found stand in no utilisable connection to the borderline between adjoining texture areas. Therefore a comparison of the author's border detection routine to traditional edge detection algorithms is obsolete.

Figs. 73(a) and 73(b) are similar to Figs. 65 and 71 with the colours replaced by texture. The borderline though is still artificial.



(a) Sharp bordered multi-texture image. (b) Fuzzy bordered multi-texture image.

Figure 73: Colours of Figs. 65 and 71 replaced by texture. This is how a multi-texture image looks like.

More information on the subject of segmentation can be found in [Gol96, HD94, LF93, Law80b, OQ94, US96].

3.6 The Texture Classification System in WiT

Showing all the Sub-iGraph and modules that build the Texture Classification System in WiT[12] is not necessary to understand the principle. Instead some part will be presented, that show the structure and the principles of graphical programming.

[12]For general information on the WiT Image Processing Software Tool please refer to Appendix C

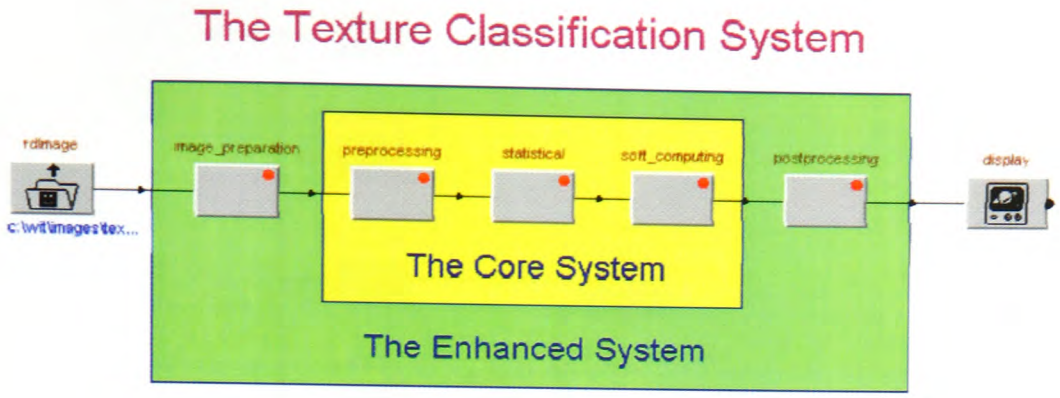


Figure 74: The WiT iGraph of the main modules of the Texture Classification System. Only the central parts, CS and ES, are being displayed.

Fig. 74 shows the top layer, i. e. the main iGraph of the Texture Classification System in WiT. The BDR and AOS are not shown for clarity.

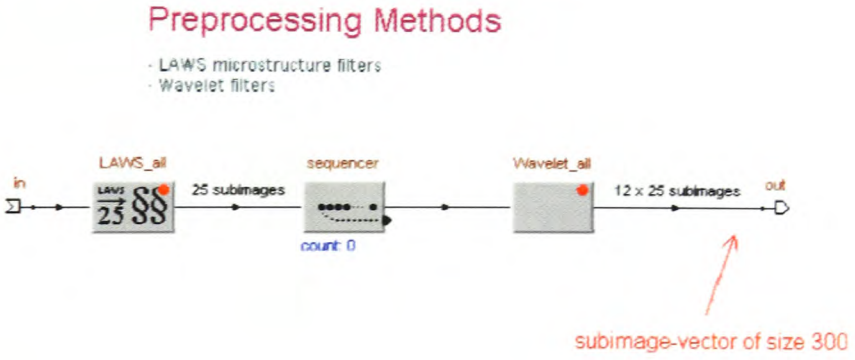


Figure 75: The WiT sub-iGraph of the LAWS and wavelet modules.

Fig. 75 shows two of the filtering and transformations methods, the LAWS micro structures and the wavelets module. The sequential connection can be observed. The other preprocessing methods are included in the same fashion. Fig. 107 gives an overview.

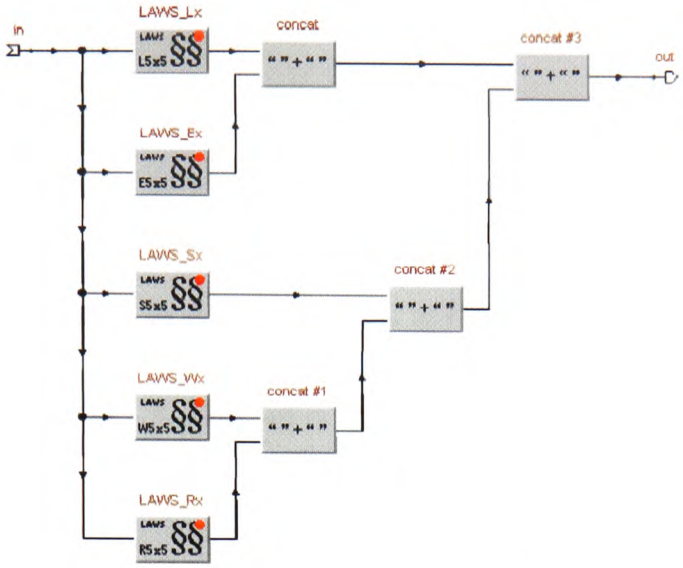


Figure 76: Sub-iGraph *LAWS-Main.*

These operators convolve the image with the LAWS micro structure matrices (Fig. 76). The sub-iGraphs hold the actual operators with each having a different first vector. Fig. 77 shows the L5 matrices.

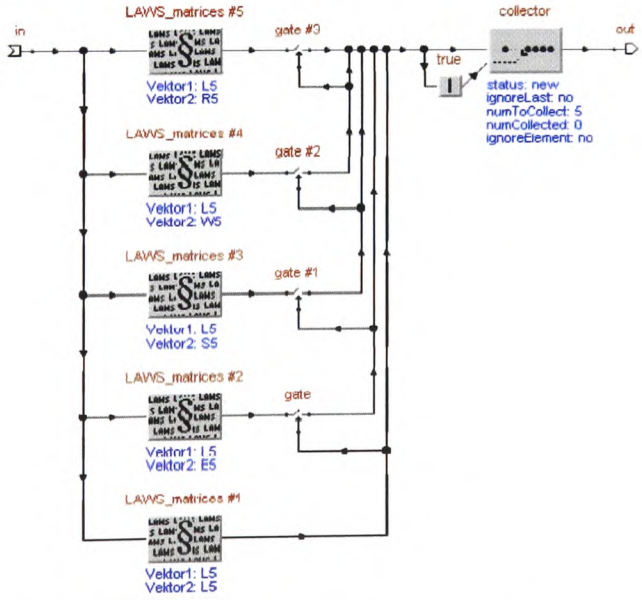


Figure 77: Sub-iGraph *LAWS-L5x5.*

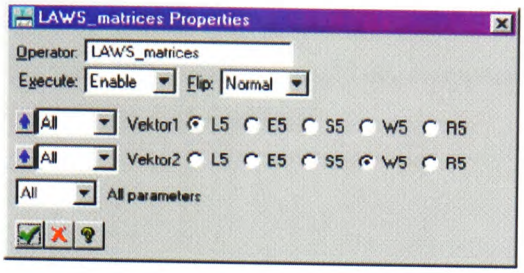


Figure 78: The properties panel of the LAWS operator.

The properties panel of the operator is used to adjust the parameters. Fig. 78 shows the LAWS operator set to L5W5. All properties panels allow manual adjustment or from within the system.

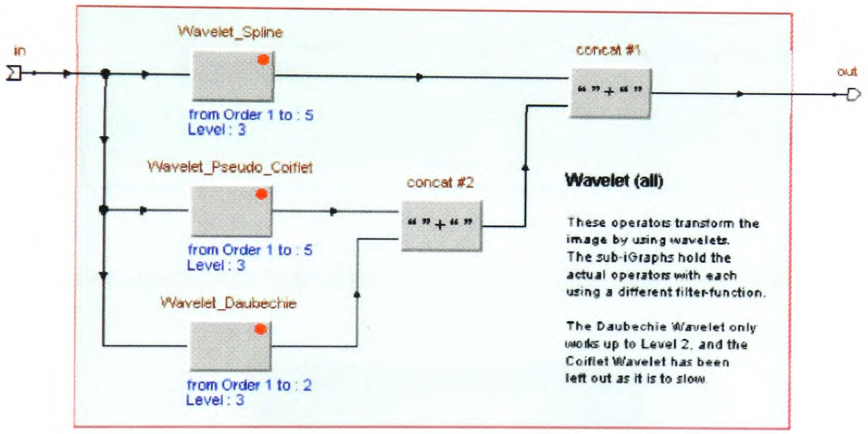


Figure 79: These operators transform the image by using wavelets. The sub-iGraphs hold the actual operators with each using a different filter-function.

The second of the preprocessing methods shown here is the wavelet module. Fig. 79 shows the main sub-iGraph for the wavelet filters. The Spline, Pseudo-Coiflet and Daubechies wavelets are used with orders one to five or one to two respectively. The level has been set to three.

Fig. 80 shows for the example of the Pseudo-Coiflet wavelets that for this setting from one input image five wavelet transformed images are being produced, each of which will later be cut into ten residues.

The properties panel of the wavelet operator shows the possible parameter settings. Level and order can be adjusted between one and ten, implemented mother wavelets are Spline, Pseudo-Coiflet, Daubechies and Coiflet[13]

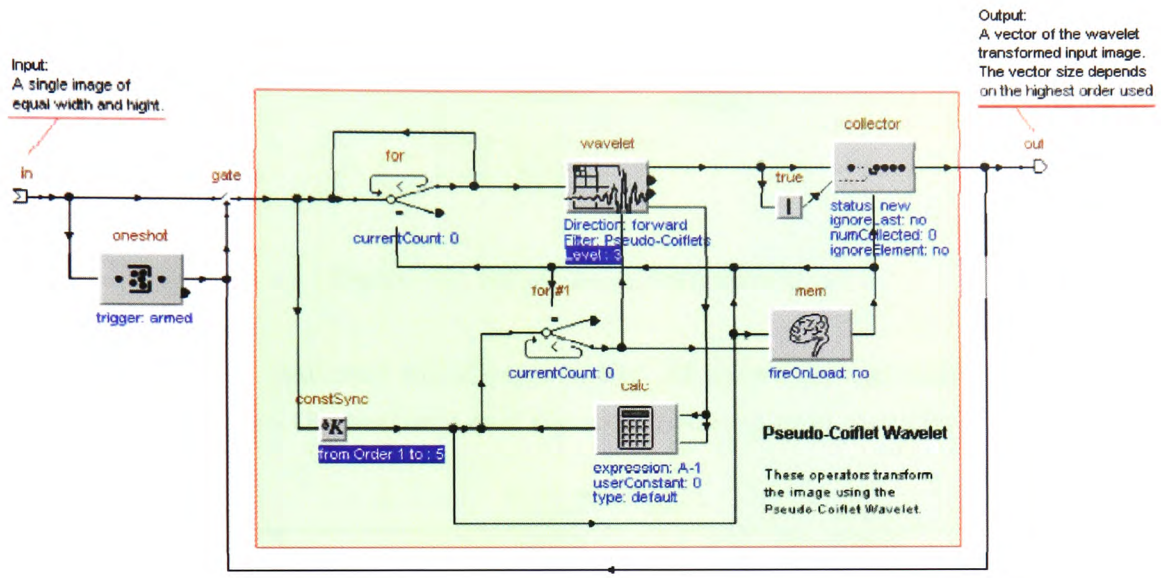


Figure 80: These operators transform the image using the Pseudo-Coiflet Wavelet.

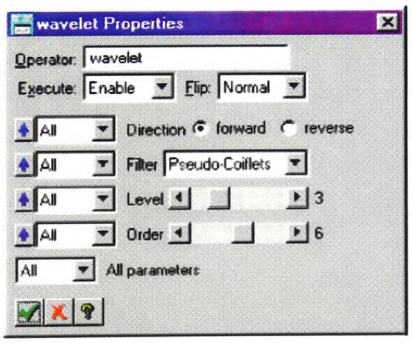


Figure 81: The properties panel of the wavelet operator, adjusted to *Pseudo-Coiflet.*

The preprocessing methods are being followed by the statistical evaluation. Fig. 82 shows the main sub-iGraph with the 1st order statistics and the Wide Cooccurrence Matrix (WCM) for the first, second and third rank.

[13]The implementation of Coiflet is rather slow compared to the others, therefore is has not been used.

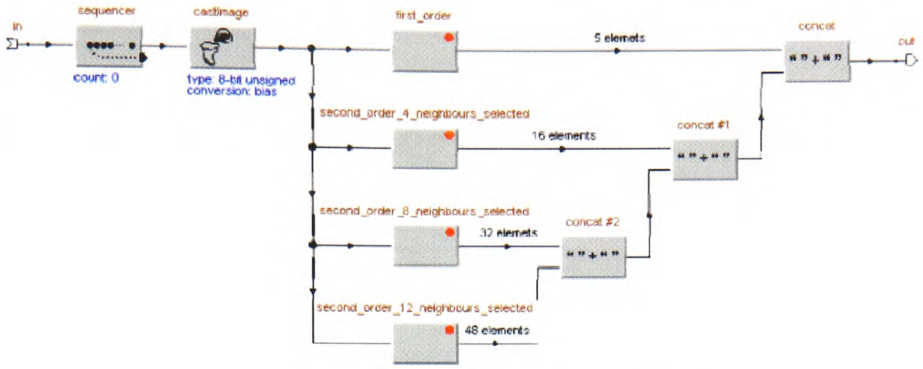


Figure 82: Sub-iGraph *Statistics-Main.*

The first order statistics sub-iGraph in Fig. 83 shows the operators used. A choice can be made between the variance and the root-mean-squared operator.

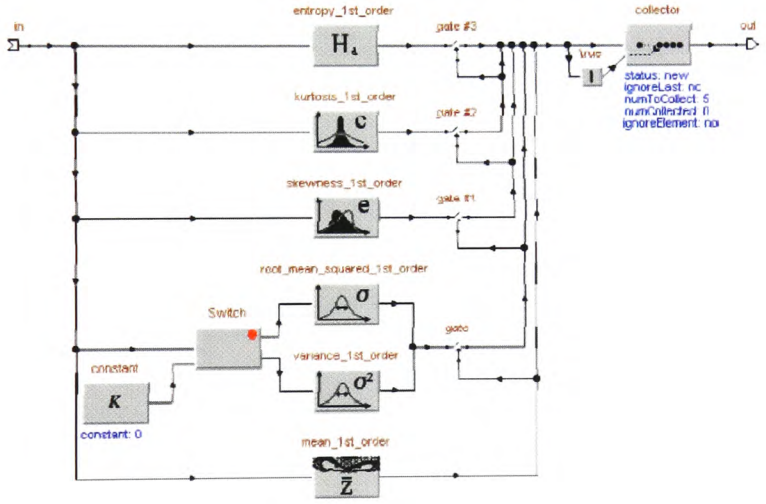


Figure 83: Sub-iGraph *Statistics-First-Order.*

Figs. 84, 85, and 86 show the use of the WCM with preset neighbours. Each of the modules hold the sub-iGraph shown in Fig. 87. At that level the cooccurrence matrices are being calculated according to the parameter settings and the statistical properties are being evaluated.

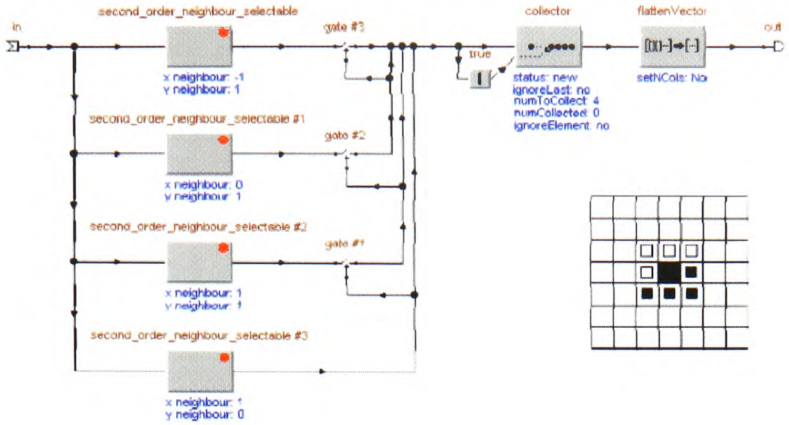


Figure 84: Sub-iGraph *Statistics-WCM 4 Neighbours.*

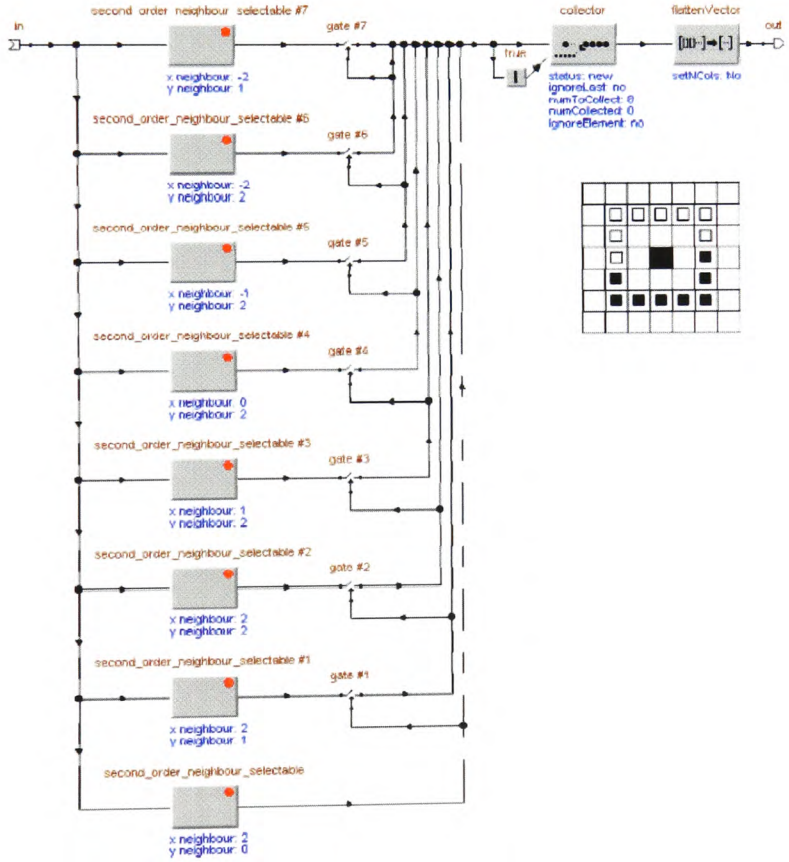


Figure 85: Sub-iGraph *Statistics-WCM 8 Neighbours.*

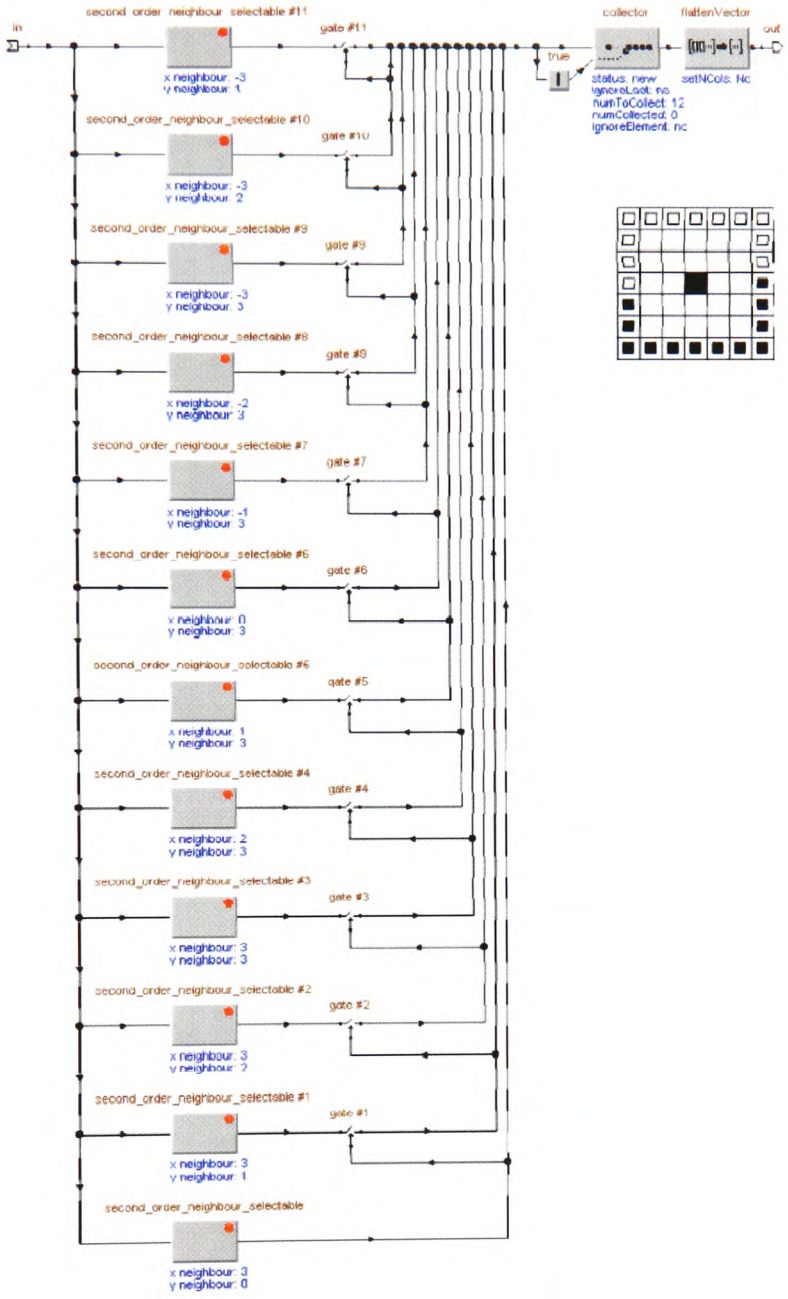


Figure 86: Sub-iGraph *Statistics-WCM 12 Neighbours.*

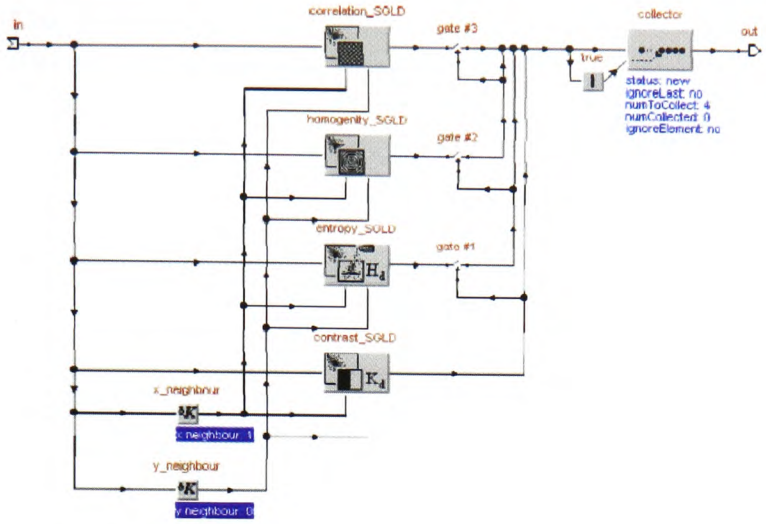


Figure 87: Sub-iGraph *Statistics-Cooccurrence.*

The properties panel in Fig. 88 shows that of the contrast calculating cooccurrence operator with a setting for the neighbour [10;7].

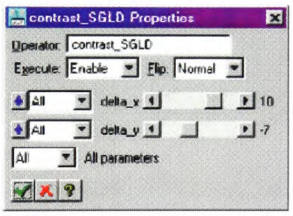


Figure 88: The properties panel of the WCM operator for contrast calculation.

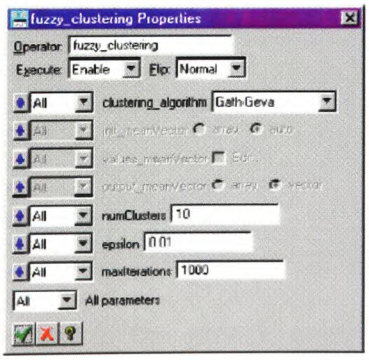


Figure 89: The properties panel of the fuzzy clustering operator, adjusted to *Gath-Geva.*

Finally the Fig. 89 shows the properties panel of the fuzzy clustering operator. Here it is adjusted to the algorithm by GATH and GEVA. Alternative settings are the fuzzy-c-means algorithm, the GUSTAFSON and KESSEL algorithm and the sharp k-means clustering algorithm.

The properties panel of the genetic algorithm operator holds a great number of parameters that can be adjusted. Fig. 90 shows the main properties panel, whilst Figs. 91, 92, and 93 are sub-panels.

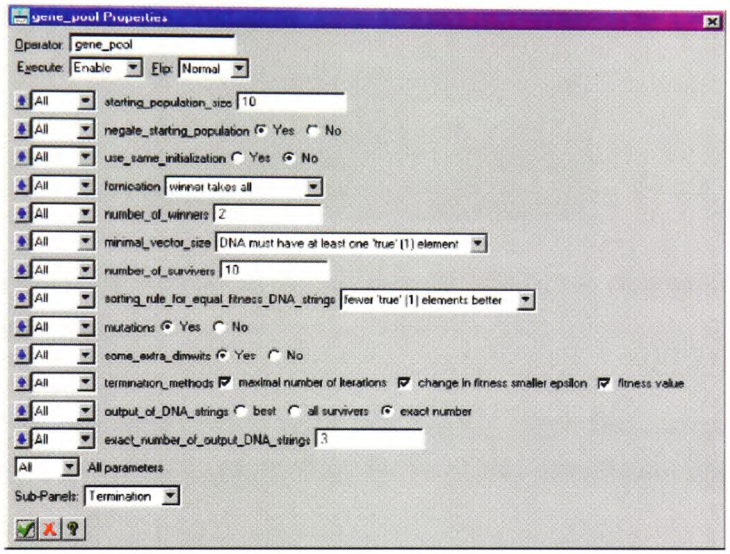


Figure 90: The properties panel of the genetic algorithm operator.

The functionality of the operator has already been described in Section 2.8. It might be interesting to note that up to three different termination methods can be combined.

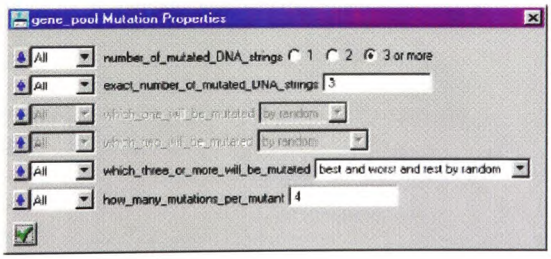


Figure 91: The properties sub-panel of the genetic algorithm operator for the setting of the mutation parameters.

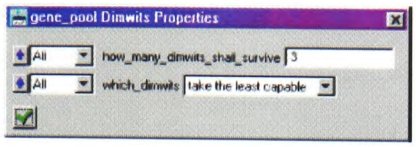


Figure 92: The properties sub-panel of the genetic algorithm operator for the setting of the dimwits parameters.

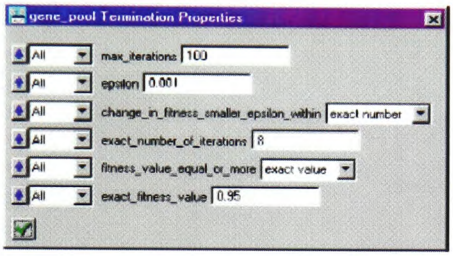


Figure 93: The properties sub-panel of the genetic algorithm operator for the setting of the termination parameters.

Fig. 94 shows a test application were the genetic algorithm operator is used for an optimisation task. The sub-iGraph in Fig. 95 produces the fitness graph shown in Fig. 96.

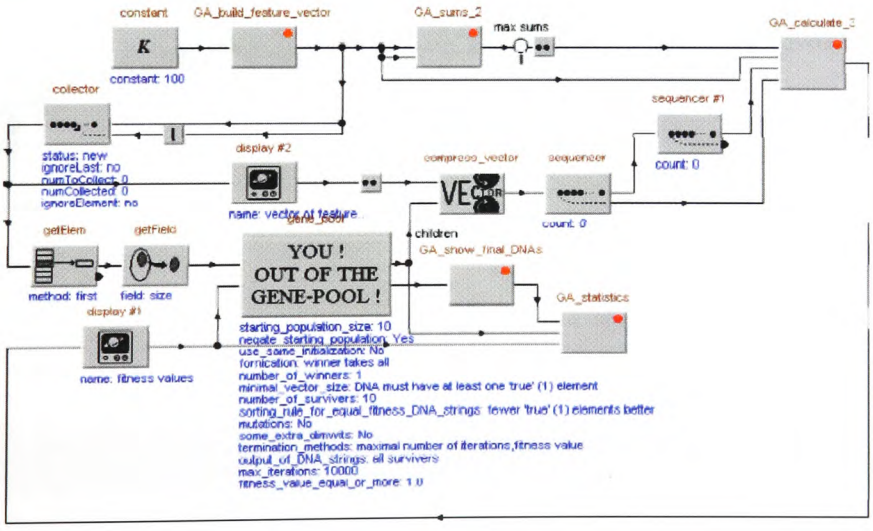


Figure 94: A test application for the genetic algorithm.

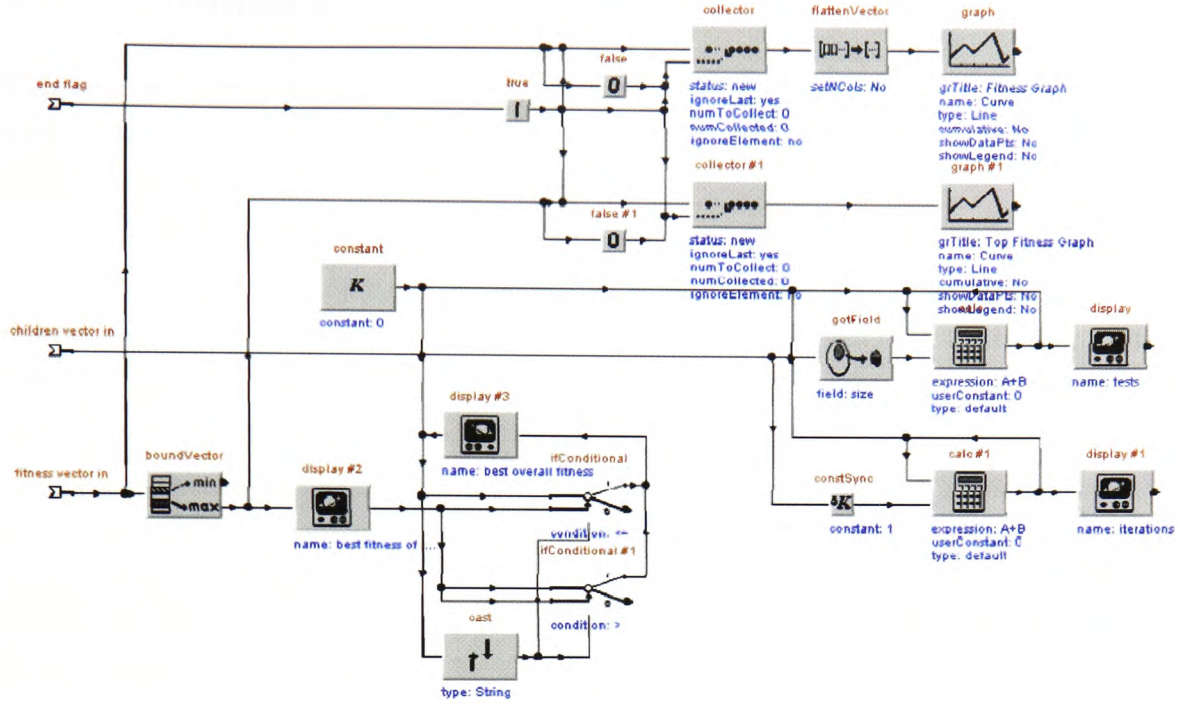


Figure 95: The statistics sub-iGraph of the test application for the genetic algorithm.

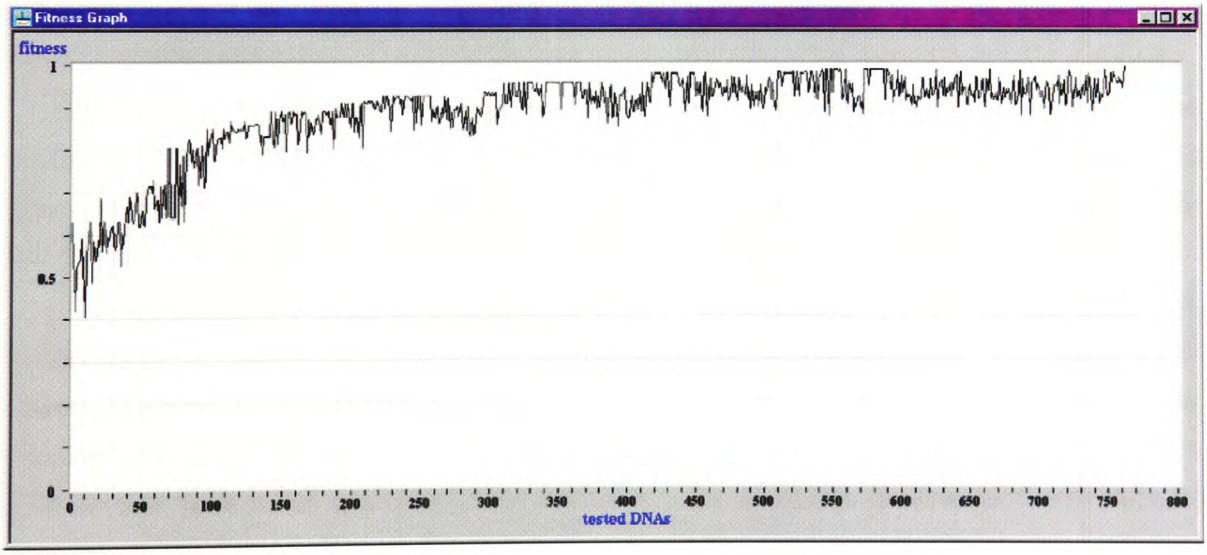


Figure 96: A typical fitness graph produced by the AOS.

3.7 Summary

This chapter introduced the author's new design and construction of the Texture Classification System. It consists of the Core-System, the Enhanced-System, the border detection routine and the automatic optimisation module.

The Core-System performs the actual texture classification. Image processing modules as described in Chapter 2 are used to extract the necessary from the unnecessary information in an image. The interim result, or rather results, are thereafter statistically evaluated. This results in a rather large feature vector which is classified by either clustering or neural networks.

The Enhanced-System handles larger images that may consist of more than one texture. The image is being prepared for the classification by the Core-System. Additionally post-processing routines can be included to increase the usability of the Core-System's output. This could for example be an artificial intelligence module, which has not yet been implemented, or the border detection routine.

As the Enhanced-System divides the input image into subimages in a chess-board like manner, it is possible that some of the subimages consist of more than one texture. This means that there is a border between textures within those subimages. The border detection routine describes how to detect the inter-texture-borders with a greater detail than the subimage size.

The automatic optimisation module can be used in many places of the Texture Classification System. The module consists of a genetic algorithm and a fitness evaluation. These two parts together are capable to optimise a system, whose parameter space is far too large to be completely tested or optimised manually.

The design of the Texture Classification System shows that a rather complex system is necessary to classify textures without compromising in certain areas. This again leads to two conclusions concerning the usability of the system. Today's computers are not yet capable to handle the amount of data produced by the system when used in all its depth. This means that a full scale test that includes a far reaching optimisation will only be possible in the future. But as the system has been designed very flexible it is possible to use subsets of the system for special tasks. Especially the central idea of using a system consisting of interchangeable modules has been used in a number of projects that are being described in the following chapter.

This chapter described the author's central ideas for the Texture Classification System and subsequently an artificial vision system. The necessity for a complex, modular and flexible system has been pointed out and the design and construction details, as devised by the author, have been explained.

Chapter 4

Applications

4.1 Objectives

This chapter describes the application of the Texture Classification System and the use of the design principles for real-world projects. The objective is to verify the ideas that led to the development of the Texture Classification System.

Here the design, construction and implementation of the Texture Classification System and its modules, as described in the previous chapters, are tested against reality. As mentioned before, today's computers are not yet capable to handle the full scale system. It will probably take another decade before the processing power and storage capabilities needed are available for laboratory and industrial applications of the full Texture Classification System.

The applications shown here use subsets of the full system. A manual pre-selection was made by the author to fit the system to the task at hand. This still gives a good insight into what the Texture Classification System is capable of, especially compared to traditional image processing, which does not provide solutions for the tasks described below.

The following section (4.2) shows a system for fault detection in glass containers. The initial problem with glass is its optical properties.

In Section 4.3 veneer is being tested. The properties of veneer are those of a textured surface, whilst defects in the veneer will show up as their features are out of range.

Section 4.4 describes a system for the detection of red hot steel blocks. Here the object's surface is very similar to its background, both having a rather smooth texture.

The last project described in Section 4.5 describes a setup for hair-gloss measurement. The differences between the samples are very small. Here the system needs to be very well tuned to detect and classify the differences.

4.2 Quality Control of Glass Containers

This project utilised parts of the Texture Classification System to design a quality control system for glass containers. During production of glass containers a number of faults can occur. Namely these are artefacts, air-bubbles, cracks, and distortions. Most often such faults occur during start up of the glass blowing process, or when the moulds are worn out. The goal is to identify these faults automatically.

Figure 97: Some of the bottles from the glass project.

This project was funded by the Innovative Projects Group (AGIP) of the Lower-Saxony Ministry for Science and Culture and included a collaboration with industrial partners.

4.2.1 Case Description

4.2.1.1 The Problem

The standard in the glass manufacturing industry is to perform the quality control manually. As this is very cost intensive and tiresome for the worker, cheaper and more reliable

systems are needed. Additionally the production speed has increased to a pace which makes it impossible for manual inspection.

As product quality and product safety demands a fault free output a 100% control is needed. Whilst bubbles and artefacts might only look bad, cracks lead to exploding champagne bottles. Cracks are often being detected by using high pressure air or brackets. By this test faulty bottles break up and often block the machine. These mechanical tests do not detect the dangerous 'monkey swings', which can leave broken glass particles inside a bottle. The nature of the occurring faults and the difficulty in detecting them results in the need for a quality control system.

Figure 98: A so called *Monkey Swing*, a glass thread inside a bottle.

4.2.1.2 The Task

The task was to find a solution for the detection of the occurring faults. In this project the principles for such a system had to be designed.

A prototype hardware and software set-up had to be developed which includes the principles and is easily adaptable to special real-life tasks.

It was clear from the start that hardware limitations and expected amount of data either allows only a rather simple solution or a more complex and versatile but slower approach.

4.2.1.3 The Solution

The input data, images of the bottles, is obtained from a line-scan camera via PCI-bus frame grabber. This enables high resolution images and no horizontal angle distortion. The bottles are background lit by a field of high-frequency fluorescent tubes.

For the laboratory experiments a turntable was built, that resembles the conveyor belt of a factory production line.

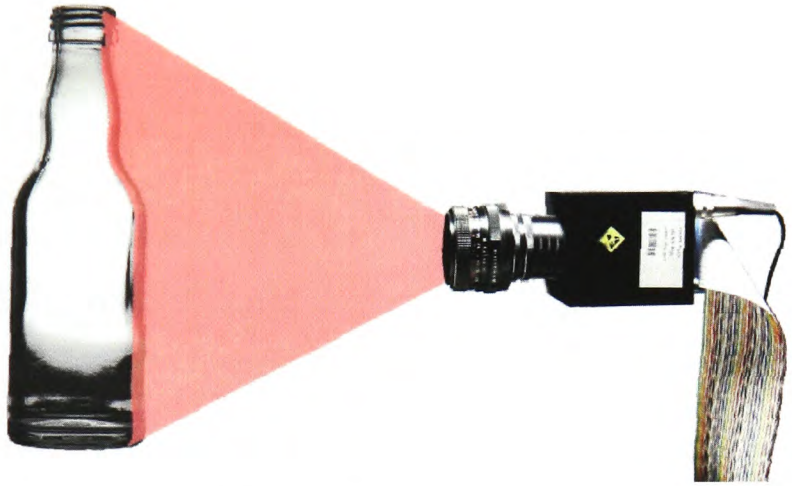


Figure 99: Principal set-up with line-scan camera.

The correct illumination of the test objects was of high importance. As glass is translucent, so are the faults in the glass, with the exception of artefacts, which are normally small stones. To be able to see the faults, the optical properties of the glass had to be taken into consideration. The glass works like a lens, projecting the background onto the sensor. For fault-free bottles the illumination system was designed to project the lit area. In the case of faults the refraction changes, and parts of the not lit background, blacked for better contrast, is projected. This can then be detected.

A major problem, as in most industrial vision systems, was the illumination system. Fluorescent tubes use alternating current, at 50 Hz, which is good enough for the human eye (cf. page 15). The line-scan camera has a pixel-frequency of 2 MHz, which means that on the resulting image nothing but black and white stripes were visible. Using high-frequency tubes reduces this effect to a tolerable level. Grey ripples overlaid the images,

but the faults in the test objects were visible. An additional routine that subtracted the background produced clearer images.

Before starting to look for faults in the bottles the search area has to be determined. Only the middle part of the bottle shall be used, as towards the sides the light refraction is too great. To check the bottle completely a second camera has to be used, set-up at a 90° angle. Edge detectors are used to locate the bottle in the image. Thereafter the image is masked to leave the relevant part.

type of fault	rate of detection	grade of difficulty	problems/remarks
major fault	100 %	low	–
crack between bottle neck and body	100 %	high	illumination and angle of view
thickening in the bottle neck	100 %	low	–
spot thickening in the bottle body	100 %	low	–
extensive thickening in the bottle body	100 %	medium	–
artefacts	100 %	medium to high	image resolution
air bubbles	100 %	medium to high	–
small surface distortions	not possible with this set-up	very high	image resolution and specialised illumination

Table 18: The rates of fault detection according to the type of fault.

Statistical methods as described in Section 2.5 were applied, resulting in feature vectors of 101 elements, that describe possible faults of the tested bottle. These feature vectors were classified using two different fuzzy clustering methods, the fuzzy-c-means and the Gustafson-Kessel algorithm.

To speed up the classification the Automatic-Optimisation-System (AOS) was used to optimise the choice of features for the feature vector. After as few as 500 generations optimised results were obtained.

The output in this prototype set-up is biased towards statistical evaluation, e. g. type of fault, position, size. In a real industrial implementation this information might only be of marginal interest. More important is a control signal that drives a machine which again removes the faulty bottle from the production line.

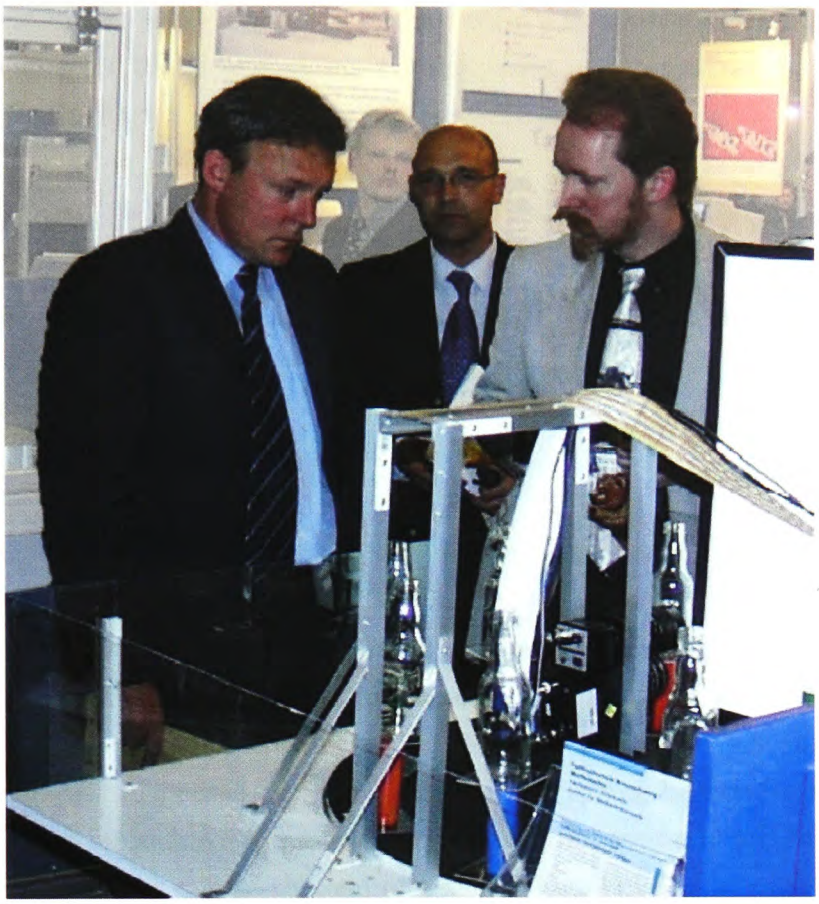


Figure 100: The glass project presented at the Hanover Fair (Hannover Messe Industrie) in 2001. In the foreground the turn-table and camera set-up can be seen. The persons are from left to right the Minister for Science and Culture of Lower-Saxony, Thomas Oppermann, the vice-president of the Fachhochschule Braunschweig/Wolfenbüttel, Prof. Dr. Winfried Huck, and the author.

4.2.2 Discussion of Results

Table 18 gives the rate of detection for the different fault classes. Even though the results are very good, this does not yet mean that the system is perfect and usable in a manufacturing surrounding. These are results obtained under laboratory conditions. Additionally the number of test bottles was with 400 pieces rather low. Some of the errors, e. g. artefacts, occurred fairly seldom, the rare 'monkey swings' in this set never.

The goal of this project was to point out possible problems for such a quality control system and show potential solutions.

The interest in this work was tested at the world's largest industrial fair in Hanover. Visitors, not only from the glass making industry, were very interested in the idea to use a complex self-optimising system for quality control tasks.

4.3 Quality Control in Veneer

Identifying faults in wooden surfaces was the objective of this project.

4.3.1 Case Description

4.3.1.1 The Problem

A mayor product in the timber industry is veneer. A good example can be seen in Fig. 101. As wood is a natural product it contains a number of flaws that are in most cases unwanted and have to be detected during the production process. Among these shortcomings are checks (Fig. 116.11), knottiness (Fig. 116.2), knot holes (Fig. 116.3), burn marks (Fig. 116.6) and discolouring (Fig. 116.10). In this project not the sheets of veneer themselves are inspected but the final product, spring boards for slatted frames. Therefore two more faults have to be detected, namely not properly applied coating (Fig. 116.9) and missing veneer due to the end of a sheet (Fig. 116.8).

In the timber processing industry the spring boards are manually inspected. This is of course cost intensive and very tiring for the worker. As both sides have to be inspected at an approximate rate of one board per second some minor faults can easily slip through, affecting the overall quality of the production. Therefore a system is needed that will automatically inspect and classify the spring board surfaces.

4.3.1.2 The Task

This project had the aim to show the potentials of a quality control system for veneer and veneer products. The system had to be able to identify the above mentioned faults and to classify the board.

Whilst the principles developed in this prototype can be used for sheets of veneer in this project only the significantly smaller spring boards were tested. The reasons for this are the sheer size of the sheets (3 metres wide) which can hardly be handled in a laboratory and the processing speed needed for this size.

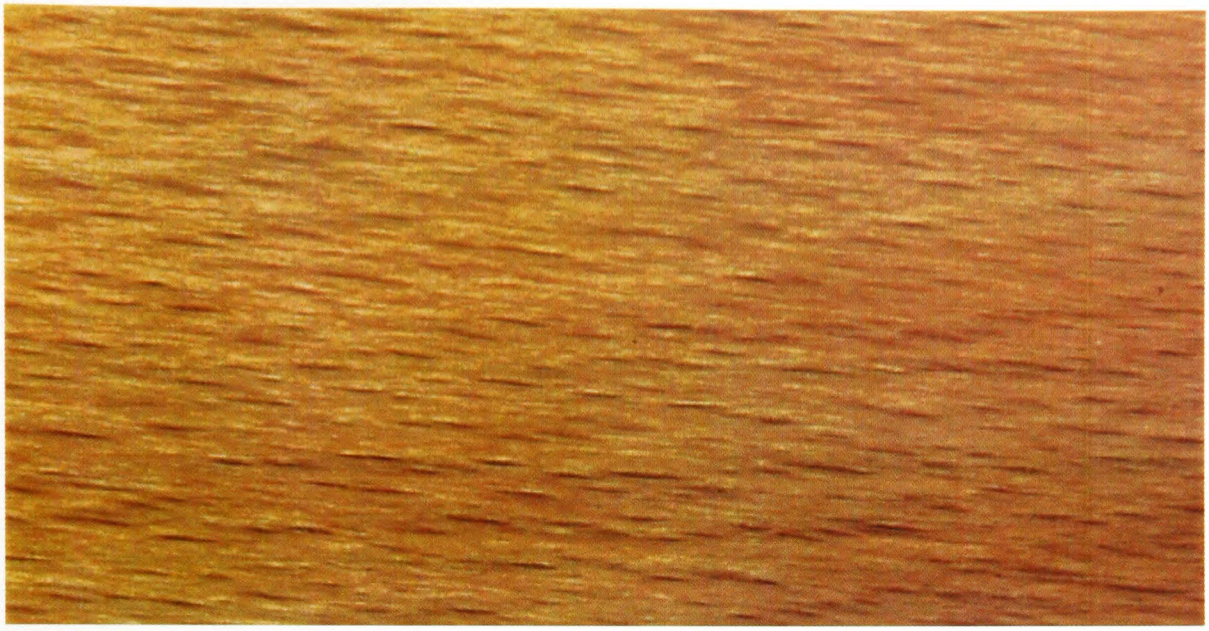


Figure 101: A veneer board for a slatted frame.

4.3.1.3 The Solution

A prototype of a system has been developed that detects and classifies the faults. A line-scan camera is used for capturing the veneer images of which the features are extracted with statistical methods. The features are classified with either fuzzy clustering methods or neural networks. Additionally genetic algorithms are used for optimisation purposes.

The sensor element is a line-scan camera attached to a PCI frame grabber. Light comes from high frequency fluorescent tubes.

By applying methods of the Texture Classification System it is possible to identifying faulty regions within a larger image. Thus endless lamina coming directly from a veneer lathe can be classified and cut accordingly.

One way of classifying the quality of the veneer is by using statistical methods for feature extraction as described in Section 2.5.

Before the statistical methods are applied some preprocessing methods are used. These aim at the reduction of redundant information contained in the images so that the relevant information can be extracted more easily.

Currently used methods are a preliminary median filtering followed by a number of image analysis methods. Among these are different kinds of edge detection, gradient extraction, SOBEL filtering, surface and spectral analysis. Next to those standard methods the images are processed through wavelets filters and LAWS-measures. Section 2.4 describes those methods in detail. The results of these computational steps are not used in the classical sense—it is for example not of interest where edges can be found—but are statistically evaluated.

Additionally to first order statistical calculations—like mean, variance, skewness, kurtosis—high order statistics are used as the neighbourhood relations are of importance. Very good results have been obtained concerning the orientation of similar grey-level pixels within an image. The cooccurrence matrices used yield potential features, among which are the energy, contrast, entropy, and correlation.

All this results in a vast amount of data, but only a fraction of the extracted features provide information that is unique to a specific fault in the veneer. Therefore the number of features has to be reduced to avoid wasting computational resources.

An early version of the Automatic-Optimisation-System (AOS) was used to reduce the size of the feature vector. The method is described in Section 3.4.

For classification fuzzy clustering was used. An alternative is to apply neural network, but this was only briefly addressed in this project.

4.3.2 Discussion of Results

As the aim was not to build a turn-key system, but to provide scientific information for the industrial partner, a laboratory system was set up.

The results of the tests were good. Using genetic algorithms to reduce the size of the feature vector proofed to be a very good and fast approach for optimisation and thus speed up of the system. For the tests 101 features were extracted with the help of the preprocessing and statistical modules. This means that the optimal subset of features has to be found among all possible $2.5 \cdot 10^{30}$ solutions. The optimisation module finds an optimal solution within 500 tests, sometimes even significantly faster.

The fuzzy-c-means and Gustafson-Kessel algorithms used will not give good results for any kind of clustering of the feature plane. In combination with the optimisation module those feature subsets were found that build clusters that are separable.

A further speed up of the system can be achieved by compiling a subsystem once the optimal setup for the individual task has been found.

Still the resources needed for a system are expensive. Especially if faults in veneer sheets, not only rather small boards, shall be detected, the speed of the system must be very high. A lathe cuts sheets 3 metres wide at a speed of one metre per second. The image resolution has to be better than 1 mm^2 per pixel. This means more than 3 million pixel, or approximately 10 MByte of data has to be processed every second. To provide a cheap system, and that is needed as most veneer producers are small companies, standard computers have to become faster[1] and fast line-scan cameras have to become cheaper.

4.4 Positioning Control of Steel Blocks in a Rotating Oven

In this project methods were explored to identify red hot steel blocks inside a rotating oven.

4.4.1 Case Description

The industrial partner in this project produces seamless steel tubes. For this process octagonal steel blocks, each weighing 90 kg, are heated to a temperature of approximately 1250° C before being push-formed into tubes.

A grasping arm places the steel blocks, four in a line, onto a turn-table inside an oven. The blocks, pre-heated to approximately 300° C when they come in, then rotate in the oven and are being heated to the end temperature. A second grasping arm takes the almost white glowing blocks out of the oven again. Fig. 102 shows the procedure.

4.4.1.1 The Problem

The problem that arises is, that the blocks sometimes roll away from their position. As the grasping arm is a purely mechanical device, the machine has to be stopped and the block has to be pushed with a stake onto its position again, for the arm to be able to

[1] Computers increase their performance by the power of two every 18 month according to MOORE's law.

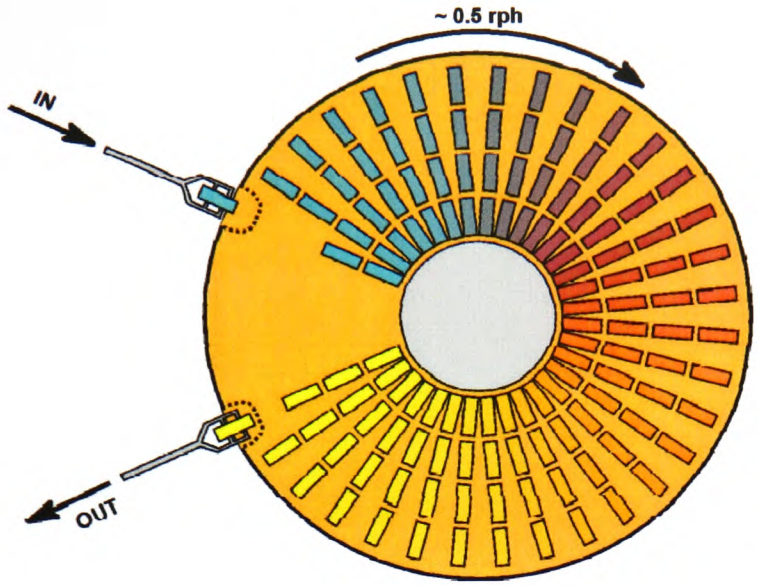


Figure 102: Sketch of the rotating oven.

grasp it. The pushing itself has to be done manually and occurs only a few times a day, but the position control, until than manned 24 hours a day, shall be done automatically.

4.4.1.2 The Task

In this project the task is to test ideas and existing algorithms for the position control application. Hereby the aim is to find rather simple but robust methods that can be used in a low profile hardware system, e. g. a smart camera[2].

4.4.1.3 The Solution

A number of problems had to be addressed in this project. The initial problem for every image processing system is to obtain optimal images. Extra illumination is not needed in this case, as the steel blocks and the rest of the inside of the oven are glaring already. The camera had to be placed 6 metres away from the oven. Any closer and the heat radiated from the blocks which are being taken out would destroy it. The field of view should not be blocked by the grasping arm. Fig. 103 shows a steel block just being taken out. The

[2]This is a camera which incorporates a small computer, often DSP based.

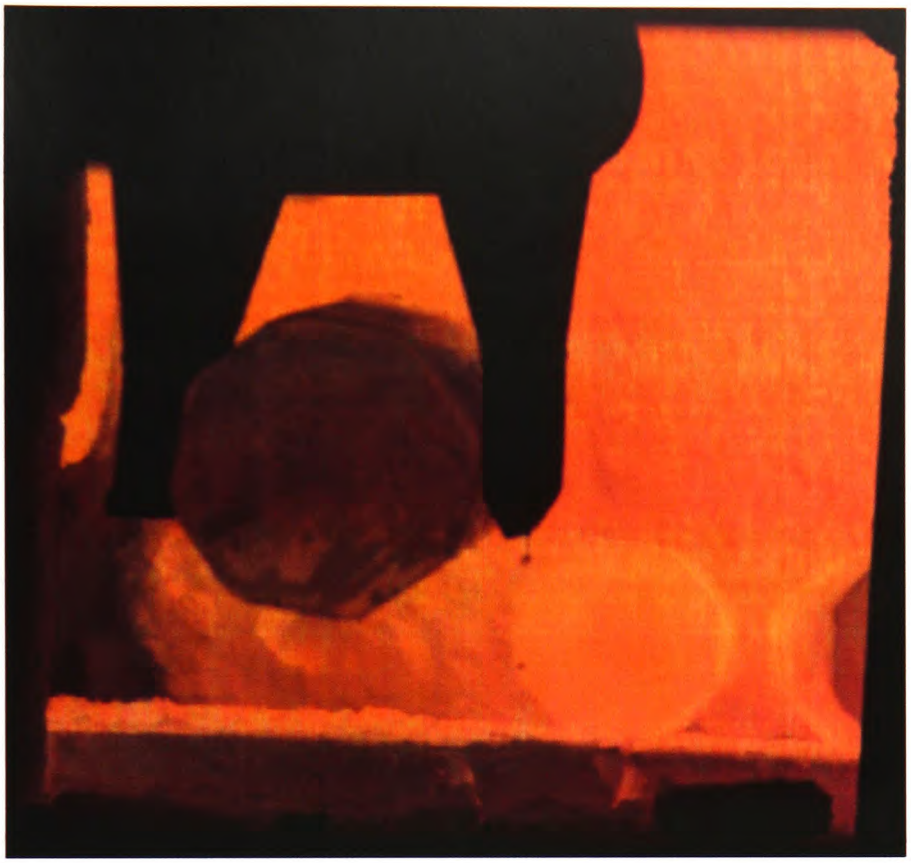


Figure 103: Steel block being taken out of the rotating oven.

block is red hot even though it looks rather dark. This is due to the very small aperture setting of the camera.

The next problem is the image quality itself. The air shimmers due to the heat blurring the images. Additionally the oven and the steel blocks have the same temperature, thus radiating the same amount of light. In the Appendix the Fig. 117 shows a number of images from steel blocks at various positions. The steel blocks in the front row shine darker, as there is less reflection of light from the oven's inner wall – the front face of a block, being made of steel, acts as a mirror.

For the image acquisition a number of different set-ups were tested. These incorporated different aperture settings with deliberate over and under exposure and the usage of optical filters that only pass through certain parts of the light (red, green, blue, and infra-red). The differences obtained by those measures were rather small, with the green filter at normal aperture setting giving the best results.

Next and more important were the tests of the image processing modules. These tests proposed that the best results can be obtained when using noise reduction filters followed by edge detectors. Whilst being rather simple this gives usable results for the proposed position control system.

4.4.2 Discussion of Results

As the background of this project was technology transfer to the industrial partners[3], many different filtering and transformation techniques were tested. Applying the Texture Classification System was not planned, as for the final set-up a smart camera had to be used. Processing power on such systems is rather limited.

The final system uses noise reduction filters and edge detectors for the position control. The industrial partner has successfully implemented a system based on the information obtained through this project.

4.5 Measurement of Hair-Gloss for Product Tests

The final project[4] mentioned in this book is about hair and hair-care products. The industrial partner in this project compares newly developed products against those of competitors.

4.5.1 Case Description

Hair strands, bleached or natural, are prepared, i. e. washed, with the test-products. They are then placed onto a carrier, a piece of tube, as can be seen in Fig. 104. These prepared carriers are placed inside a specially prepared light-box, called 'gloss-box' (Fig. 105). A panel of trained people determines the gloss factor by comparing two probes at a time.

[3] ... and teaching students

[4] There are more projects that profit from the ideas and methods of the Texture Classification System (TCS). Most important are the quality control systems for car-navigation and car-radio systems. As the texture aspect, at least at the moment, is rather small, they shall not be discussed. Another project identifies ill readable printed text, as often found on product packaging. Part of the Texture Classification System are reused, especially the neural classification modules. Results have been shown at the 2004 Hanover industrial trade fair (HMI 2004) and have been published, too. Further projects are on-going that look into aspects of driver assistance and autonomous driving. These projects used clustering for classification and will incorporate neural networks and genetic algorithms in the next development stages. Results will be published in future.

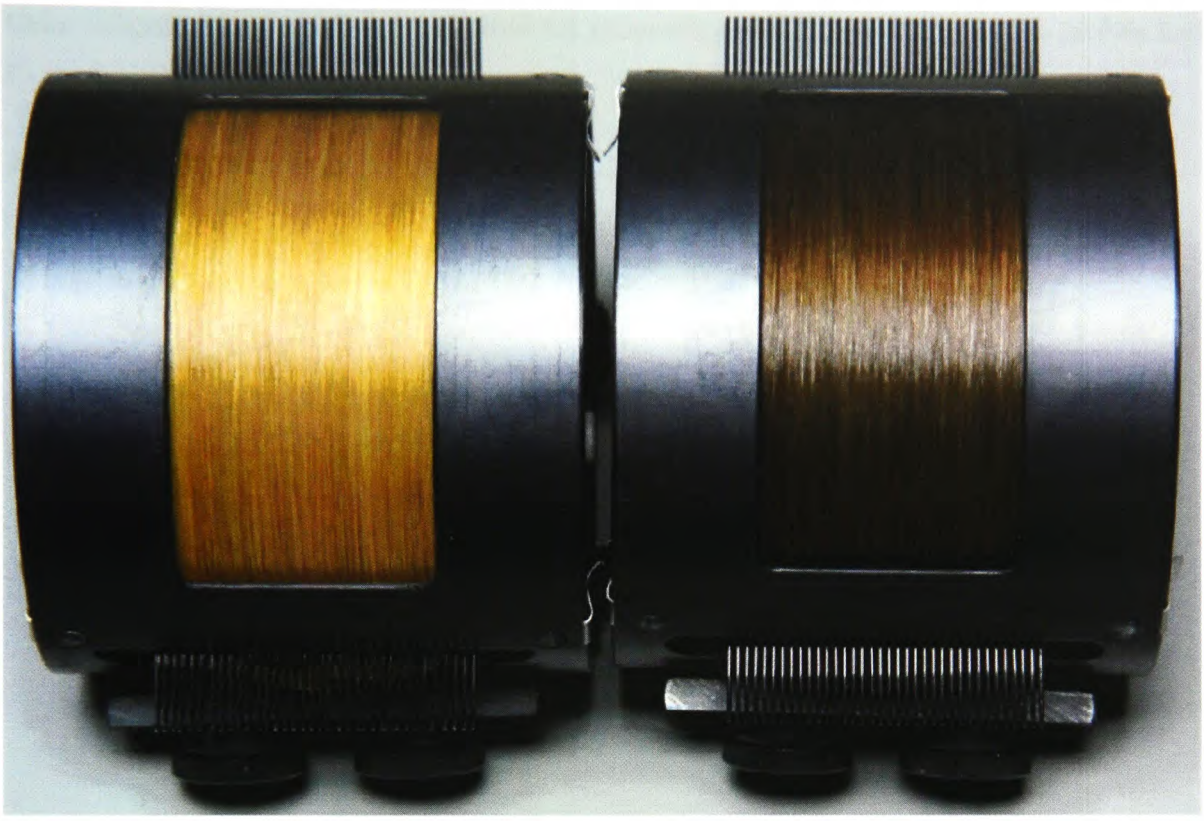


Figure 104: Hair carriers used for gloss determination.

4.5.1.1 The Problem

This project has a number of objectives. At first the test set-up had to be optimised. In the second step the members of the panel should be put into a position to judge in their own time and separated from the others. And the final goal is to replace the panel by an automatic gloss measurement system.

Prof. Kruse of the University of Magdeburg, well known for his research in soft-computing and data analysis, asked the author to participate in this project, as a great deal of image processing is involved.

4.5.1.2 The Task

To reach the final goal of automatic gloss measurement the existing 'gloss-box' had to be optimised. Thereafter the set-up for taking high-resolution and high-quality digital images had to be defined. Next a software was needed to display the images and have

them judged. Finally software routines for processing and classifying the hair probes had to be developed.

4.5.1.3 The Solution

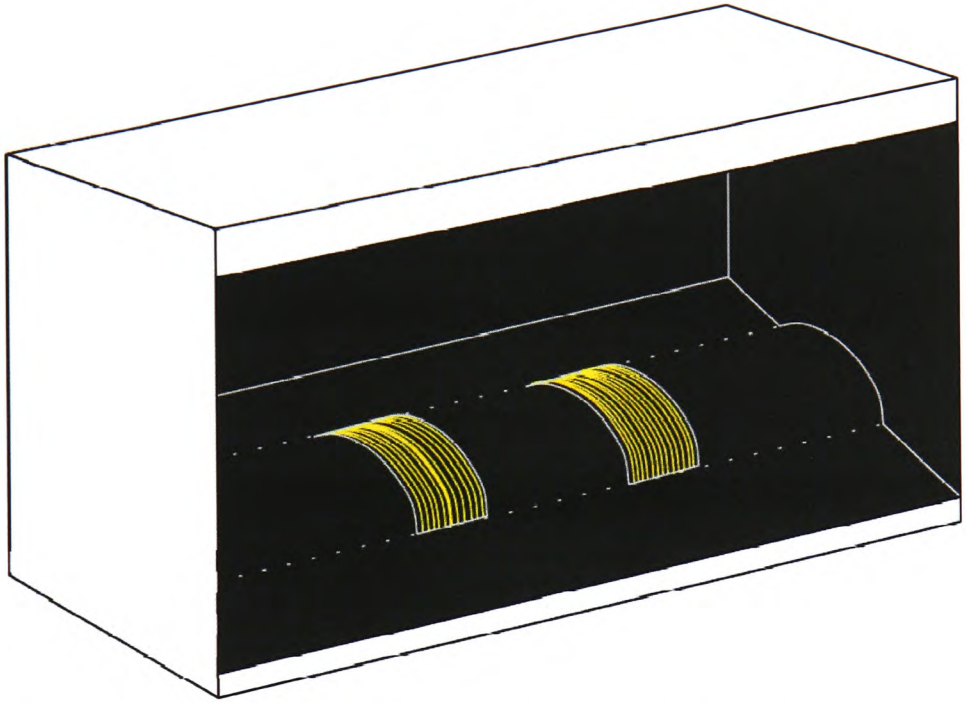


Figure 105: Sketch of the gloss-box used for the panel test and the photographies.

As can be seen in Fig. 104 the tube surface reflects more light than the hair. The design of the light-box and the light source itself was improved based on the knowledge of correct lighting and image acquisition. Fig. 118 in Section B.4 shows the results. As the output of the light source deteriorates significantly during its life-circle, measures had to be taken to allow camera and image calibration.

To enable the panel to judge the probes 'off-line' a software was created that displays high-resolution images of the probes on a monitor-screen. A calibrated ray-tube monitor has to be used to obtain useful results. This system, which is in its test phase at the time of writing, enables faster tests, and, more important, old probes can be tested against new ones. The image processing part in this part is to calibrate the images.

The automatic gloss measurement system is still under development. This is mainly due to the fact that until now only rather few pre-classified images are available. The

system itself will use some of the filtering routines described in Section 2.4 and the statistics described in Section 2.5. Classification will be done by clustering or fuzzy clustering, depending on the properties of the feature vectors. Again the Automatic-Optimisation-System (AOS) will be used to optimise the system.

4.5.2 Discussion of Results

Not many results are available so far. The feedback from the industrial partner's staff and panel members concerning the optimisation of the 'gloss-box' and the software is very positive. The performance of the automatic gloss measurement system has to be determined once enough probes have been made available.

4.6 Discussion

As already discussed in the previous chapter, today's computer systems[5] are not yet capable to handle the Texture Classification System in full scale. Therefore the design principles were used to find solutions for problems as they exist in industry.

The applications described in this chapter show that a complex vision system is needed for difficult classification tasks. The author's intention is to emphasise the fact that artificial vision is not a purely academic subject only useful once it equals human capabilities, but that the ideas, modules and especially the complex interconnectivity can be utilised for today's industrial tasks that cannot be solved with of-the-shelf solutions.

Due to the limitations of today's computer hardware all the applications mentioned in the previous sections needed a purpose build system. Using the full Texture Classification System was not an option for tasks with time limitations. But the design principles and image processing and classification modules were used. Such manually optimised systems proof the feasibility of the overall approach.

A critical review of the applications and the system used to handle them might suggest that this is like *using a sledge-hammer to crack a nut.* To a certain extent this might be true. Using the Texture Classification System of course means carrying along an

[5]Existing supercomputers would provide a hardware platform that comes closer to what is needed. Most of those computer systems are massively parallel which would be a perfect match for the Texture Classification System. Unluckily such systems where not available.

overhead of methods not needed for a very specific task. But this is only so long correct until the self optimisation of the system produced a lean, application specific system. For the moment and the applications described above this optimisation was done manually for the known reasons.

It is important to stress the point that the Texture Classification System is not a tool for a special task, but a general collection of tools with a complex control mechanism to suit all sorts of tasks. The Texture Classification System is a step towards artificial vision.

To sum it up the results showed that using a complex, modular system is a valid approach and enables solutions to problems that were not solvable with traditional methods.

Chapter 5

System Evaluation

5.1 Objectives

This chapter describes the application of the Texture Classification System for texture classification itself.

The texture classification is, at least for the moment, an academic application. The intention is to show how an important part of the long term goal of artificial vision can be solved. Here the need for computing power becomes most evident.

5.2 Texture Classification

Figure 106 shows a simplified layout of the Texture Classification System. It is composed of modules and layers. The centrepiece is the Core-System (CS) which consists of three modules as described in Section 3.2. The CS classifies the input texture images and is itself the centre module of the Enhanced-System (ES) which is described in Section 3.3. The ES is capable of identifying single texture regions within a larger multi-texture image. The Automatic-Optimisation-System (AOS) is made up of the genetic algorithm module and the fitness evaluation module as described in Section 3.4. This subsystem can be used for optimisation tasks throughout the complete system. An addition to the CS is the Border Detection Routine (BDR) which is used to control the choice of subimages at the input to obtain texture borders at the output. This routine is described in detail in Section 3.5.

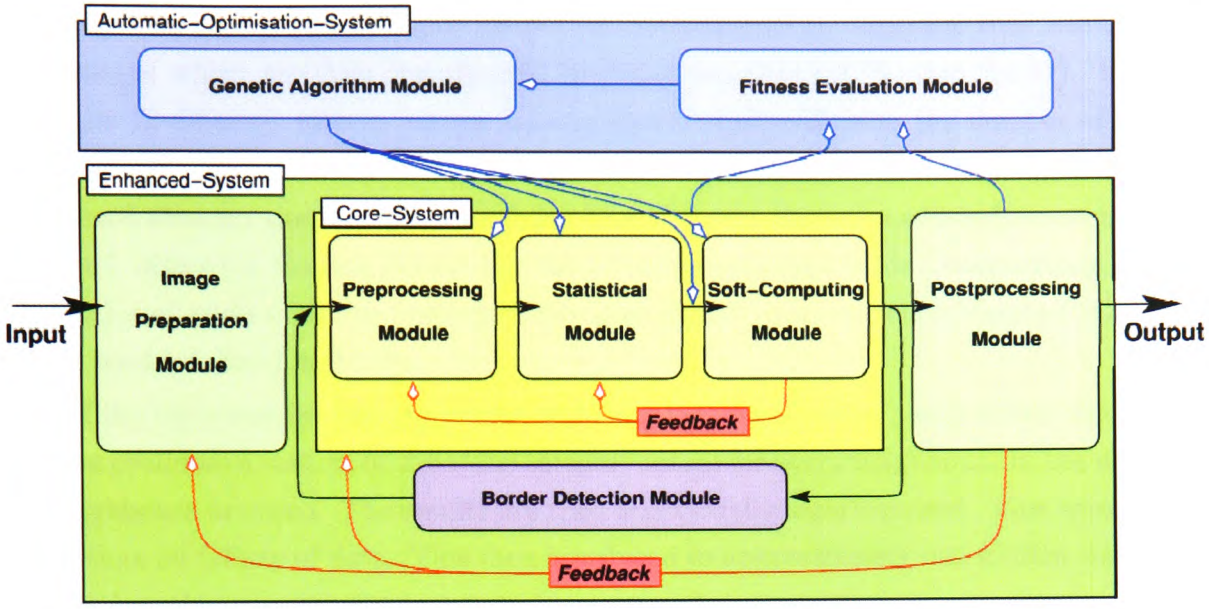


Figure 106: The Texture Classification System.

5.2.1 Usage

To fully test the Texture Classification System with today's computers is not possible. The system is too complex to be handled with available hardware. Therefore the modus operandi of the system is being described.

5.2.1.1 Creation of the Feature Vector

An example will explain the how this takes place. The input multi-texture image is first cut into small subimages that shall be tested separately. Taking a 2 MPixel image[1] as input, i. e. 1600 × 1280 pixels, and cutting it to 32 × 32 size subimages will provide 2000 subimages with a size of 1 kByte each, for colour images 3 kByte respectively.

Each of these 2000 subimages, and that does not take the BDR into account, is at first filtered and transformed, as described in Section 2.4. For this example 10 different operators[2] are being used. These 10 new images are then filtered using the LAWS micro statistics. The convolution produces 25, the additional Texture Energy Measures (TEM)

[1] compared to today's consumer cameras this is a rather small size. For example x-ray images are digitised to approx. 6 MPixel, aerial photographies used to detect unexploded bombs to 30000 × 30000, which are almost one billion pixel.

[2] This is a very low number, as most filters allow different kernel sizes and layouts.

15 images for each of the 10 input images (cf. Section 2.4.12). Together that accounts for 410 images which are then transformed by using wavelets (cf. Section 2.4.11). For this example 12 different wavelet setups are employed, which will raise the number of images to 5330. At this point the first module of the CS is finished and the images are now processed through the statistical module. The 1^{st} order statistics will reduce each image to just 5 values (cf. Section 2.5.1). For the 2^{nd} order statistics Wide Cooccurrence Matrix (WCM) and additional one version of the Logarithmic Cooccurrence Matrix (LCM) are being used (cf. Section 2.5.2).

If the CMs test for 256 greylevels, which is the normal value[3], each of the 5330 input images produces a matrix of 256×256 integer[4] values for every neighbour. In the example 24 neighbours are used. The results are then additionally logarithmised. This amounts to more than 30 GByte of data. This then is reduced to approximately one million statistical values, but the amount of data has to be processed.

It is absolutely clear that there is a 'certain amount' of redundancy in those one million values that describe a texture subimage of the size of 1024 8-Bit pixels. But, at least in theory, 256^{1024} ($\approx 10^{2466}$) different images can be described in that format.

Fig. 107 shows the above described routine.

In this example only the greylevel intensity value of the texture image has been addressed. If the colour of the image has to be taken into account, additionally the hue and saturation values, or the red, green, and blue colour planes have to be handled in the same way as described above.

5.2.1.2 Reduction of the Feature Vector

This above produced feature vector is the input for the third module of the CS, where with the aid of soft-computing methods the classification takes place. Here the one million element strong vector is reduced to, in the best case, one single value representing a certain texture. If membership functions are used membership values for all known[5] textures are given. In that case the ES will choose the most appropriate texture based on the membership values and the classification of bordering subimages.

[3]Greylevels could be reduced to 4 or 6 Bit per pixel to save computational resources, but also raised to 10 or 12 Bit per pixel for higher accuracy, provided the image is digitised with that resolution.

[4]2 Byte long integer values are needed for the matrices of the subimages of this example.

[5]That means, of course, known to the system.

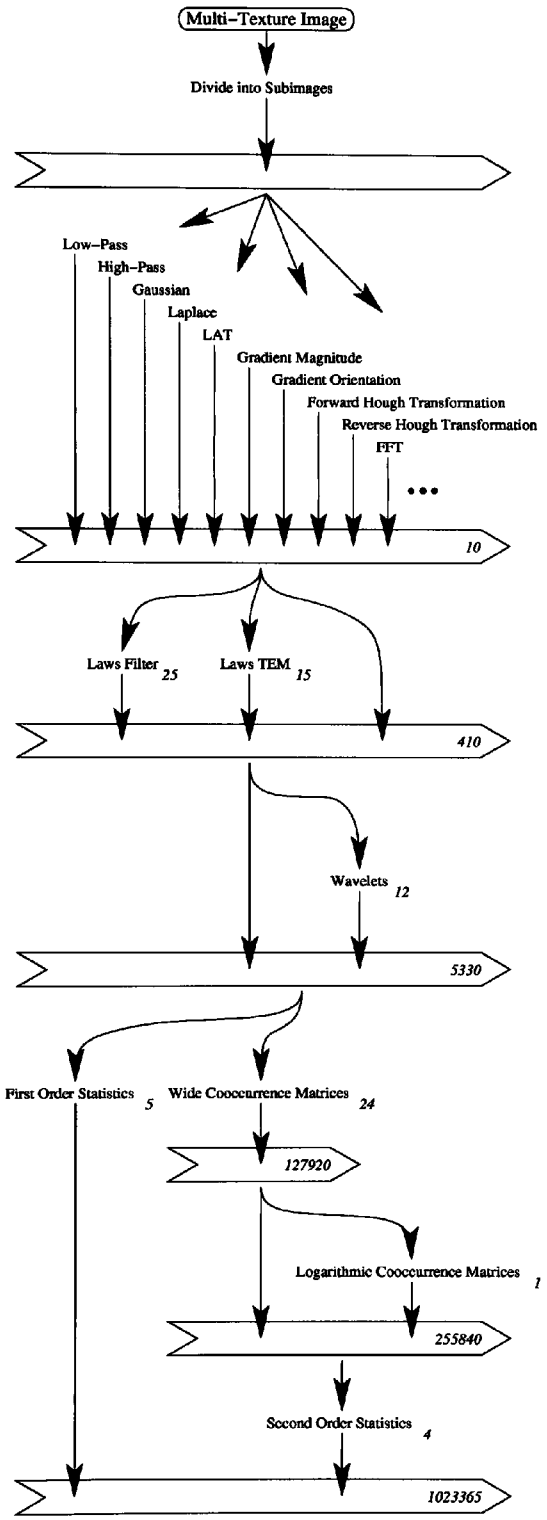


Figure 107: The feature vector.

Reducing this amount of data to just a single information is not too difficult. The human brain does it every fraction of a second (cf. Section 2.2.2). But today's computers are not yet in the position to handle a NN with one million input neurons, or a one-million-dimensional feature vector space as would be needed for clustering.

This means that, at least for the moment, only subsystem of the Texture Classification System can be used. The task of removing some of the modules has to be done manually, and not, as planned, by using the AOS.

Of course each subsystem can use the AOS to reduce unnecessary data. But it has to be said, that combining a great number of data reduced (optimised) subsystems will not make an optimised complete system. The extracted statistical values are not used for classification separated from one another but in combination with all other values. In analogy to the statistical orders the classification is of 2^{nd} order.

5.3 Discussion

The author hopes that no reader expected a working artificial vision system. It will take at least one, more probably two decades before artificial vision systems[6] become reality.

This work describes the necessary complexity on the way towards such systems. It explains that only by combining methods from many different research areas demanding image processing task will find a solution.

The choice of methods that can be included as modules into the system is, in theory, without limitation. This means that every approach for texture classification or comparable applications taken by any member of the scientific community is a valuable addition to the system. The usefulness of any new module in the systems context will automatically be established with the help of the AOS.

5.4 Conclusion

At the very beginning this research project was about automatic recognition of coarse-grained non-regular structures, later more clearly defined as texture. Whilst working on the subject it became clear that *producing one more application specific method for*

[6]Meaning a system that is equal or better than the human visual system.

the identification of a well defined set of texture images as can be found in literature in numbers was not an option for the author. A more generalised approach was desired.

This led to the development of the Texture Classification System. The system needed to be trainable and adoptable to all sorts of textures, especially those that are not known beforehand. This provoked the necessity of a universal framework that can hold and control all thinkable image processing and classification principles which at the same time has self-optimising capabilities.

The result is a system that consists of a large number of modules for image processing, feature extraction, mostly statistical, and classification. All those modules are being controlled by a system which allows automatic optimisation. As already mentioned several times the necessary computing hardware is not yet available. Therefore only the modules themselves and subsets of the system could be tested.

Chapter 4 described application that either used a lean subset of the Texture Classification System or employed the principles developed for it. Making use of the system is an ongoing process. The author's latest projects are based on his research work, too. One is about the identification of ill-readable characters, i. e. badly printed, smeared or with low and varying contrast. The other, a rather new project, is a driver-assistance-system. There is still a long way to go before such systems can be effectively used in vehicles or even drive automatically. These projects have not been described in this work, even though first installations are being used in industry. Those projects are still under development and shall only give an example in which areas the Texture Classification System or its ideas can be employed.

Chapter 6

Discussions and Conclusions

6.1 Intentions

For humans the most important way to obtain information is by seeing. In today's working life and more and more in the everyday life, too, computers take over tasks that until then only humans could do. The performance of a computer does not come near the performance of a brain, but is more than sufficient for most tasks. A multitude of sensors exist to gather information which is analyzed by increasingly complex software.

But there is one area where the human's abilities are far advanced to those of modern computer systems. That is vision.

6.1.1 The Aim of the Project

Having in mind the importance of vision and the lack of capable artificial vision concepts it is all but far fetched to design a computer vision system.

Comparing the human's visual capabilities to the processing power of modern computers makes it clear that a full running implementation is not yet possible. It will take another one or two decades before artificial vision systems equal or better natural ones.

Therefore this work looks at the concepts needed for artificial vision and designs a system for a specialised task.

6.1.2 Tasks

It is important to note that the Texture Classification System described in this work can be seen as an example or forerunner for an artificial vision system. By classifying texture a great variety of tasks and techniques have to be addressed. Additionally a system design principle has to be developed which is the backbone for complex modular visual systems like those described in this work.

The tasks at hand are at first an understanding of vision and the human visual systems, next a detailed evaluation of image processing and feature extraction, a close look at feature classification and the design of an all surrounding system which allows a modular design.

6.2 Anticipations

The first level expectation is a system designed for the classification of two dimensional texture images. Next to the system design itself an explanation of texture is needed.

The second level expectation is a modular system, which is capable of being adaptable for a variety of complex image processing tasks. This includes the possibility to easily adjust or exchange parts of the system to find an optimal solution for a special task.

The third or high level expectation is a system that shows the way towards artificial vision. The system needs to be capable to handle a degree of complexity for which the available computer hardware is not yet a match. The system design has to incorporate self-optimisation routines that make the system flexible and robust.

6.2.1 Full Scale Texture Classification System

This project describes the modules and overall design of a system used to identify texture. Texture is being regarded as an interesting subject to test the principle ideas of an artificial vision system. In contrast to the identification of geometrically describable objects texture itself hold certain fuzziness and is only statistically describable.

As the goal is not to build a contraption for the identification of a well defined set of textures the main focus was laid onto the design of a flexible framework and a multitude of image processing, feature extraction, statistical, and classification modules.

The framework uses self-optimisation routines to optimise the use, parameterisation and interconnections between modules.

6.2.2 Subsets for Specialised Projects

As today's computer hardware is not yet capable to handle the full scale Texture Classification System the principles and modules have been put to use in a number of quality control projects. These projects show the validity of the approach and additionally show a way to how complex quality control tasks can be handled using vision sensors and an intelligent software system.

6.2.3 New Design and Construction Principles

The holistic approach for the system design and the use of a multitude of algorithms and strategies from different research areas are the new design and construction principles. Ideas from image processing, statistics, soft-computing, optimisation strategies, and control mechanisms were brought together to form a complex system.

Furthermore it is important that the system is re-usable for other tasks. It has to be possible to use some of the design ideas or the complete system for different applications.

Last but not least the new definition of texture is needed as existing descriptions are too vague.

6.3 Accomplished Work

To create a complex image processing and classification system, which is task independent and thus fully re-usable and which defines future self-optimising system design is in itself a rather complex task. A multitude of different research areas have to be addressed and brought together.

The following sections give a rough overlook over the major achievements.

6.3.1 Understanding of Vision

For the creation of an artificial vision system it is not only necessary but important to understand how natural vision works. The focus was laid on the understanding of the

human visual system as this is the most complex one. Even though not all details can so far be explained by science, the knowledge of the eyes as sensory organs and the brain for the data processing and classification task is good enough to be used for the design of the system. Of course the human visual system was not a blueprint for the artificial system, as the technical equivalents function in a different way.

6.3.2 Texture

Next to the design itself a better definition of texture was needed. A new, more detailed definition was introduced to be able to differentiate between texture classes.

Additionally a new texture image library had to be collected as existing libraries have their limitations. This library will be publicly available for research and increased in size. This may create a new standard for the texture research community. A number of examples are given in Fig. 114 on page 162ff.

6.3.3 Image Processing

Images hold a great amount of information, regardless of how they have been captured. This data has to be processed and features, which describe the image contents, have to be extracted. The raw image data itself is very often not useful for further evaluation, but statistically extracted information from filtered or transformed images is needed. As no universal algorithm exists for this task, a great variety of filtering and transformation algorithms were implemented and used as modules together with the modules for the statistical evaluation. All those modules together produce a feature vector (cf. Fig. 107) that describes the image contents—the texture—and which is being used for the classification.

6.3.4 Classification

There are a number of different ways of how to classify data. In this project the soft-computing approach was chosen. In that field two different methods exist, namely the fuzzy-clustering and neural networks.

Clustering and its fuzzy specialisation use neighbourhood properties in the n-dimensional data plane to find groups of data with near similar features. A number of approaches and definitions of *neighbourhood* have been discussed.

The alternative to clustering is using neural networks. A detailed introduction to the principles, capabilities and arising problems describe the potential of this approach. This has been discussed under the light of texture classification and the more general artificial vision aspect.

Both classification approaches are feasible but especially the neural networks demand high computing power and both need a large amount of pre-classified training and validation data.

6.3.5 Automatic Optimisation

As the full scale Texture Classification System with all its modules is all but a lean system optimisation routines were needed. For this task genetic algorithms were chosen. This concept is flexible and can easily be put to use for a variety of tasks which range from finding optimal parameters, choosing the best matching modules to reducing the size of the feature vector and thus enhancing the speed of the system.

The design and parameterisation of the genetic algorithm and its accompanying fitness function is being described in general and for the use in the Texture Classification System.

6.3.6 Validation

As already mentioned computer hardware is not yet a match for the full scale Texture Classification System. Therefore the design principle and the modules were tested with manually reduced subsets of the system. To show that the principal approach towards artificial vision works real life tasks were chosen. These are quality control task that could not be handled by a computer system before. These examples additionally show the need and necessity for complex computer vision systems in industry. It was about time that the step from rather simple industrial image processing to complex visual quality inspection systems was taken.

6.4 Achievements

Hidden in this work are a number of smaller and larger achievements which in combination provide a system that gets science closer to real artificial vision.

6.4.1 The Texture Classification System

The work focused on the principles and implementation of a system for the identification and classification of texture. To reach this task a variety of research areas had to be addressed and brought together to build a complex yet flexible system.

It is important to understand that the Texture Classification System is not a specialised system solely for the identification of a well defined set of texture images. This is not the case and not the intention of this project. A system has been designed that is capable to handle *visual problems*. This system can be tuned—and once the hardware is fast enough is self-tuning—towards a certain task. Texture classification was chosen as this is a complex but still manageable field which in itself is an important part for artificial vision.

6.4.2 Specialised Projects

To prove the systems adaptability a number of quality control tasks were projected and implemented. As these systems only use the principles and *hand selected* modules of the Texture Classification System they are small enough to be used with available hardware.

These projects, looking at glass, veneer, steel, and hair, handle objects that are in principle texture-like but with a smaller variety. The outcome of the projects show that the principles perform as was expected.

6.4.3 Principles

The most important part of this work is the holistic approach for the system design. The use of a multitude of algorithms and strategies from different research areas is the base of the design. Ideas from image processing, statistics, soft-computing, optimisation strategies, and control mechanisms were brought together to form a complex system.

Just as important is the re-usability of the Texture Classification System for other tasks. Some of the design ideas or the complete system can easily be used for different applications as the examples showed.

6.5 Discussion

6.5.1 Intentions vs. Accomplishments

At the beginning the intention was to identify coarse-grained non-regular structures in images. As that in itself is a rather fuzzy definition for a task, it boiled down to texture images. Looking at existing projects described in literature it became soon clear that the system developed for this Ph. D. project should be more than another highly specialised algorithm which is perfect for a good benchmark test. Thus arose the idea to design an artificial vision system.

The work resulted in the Texture Classification System which can be seen as a first instance of an artificial vision system. Due to hardware limitations it was and will be for quite a while not possible to get the full system running. Only manually reduced subsets are small enough to be handled. A number of smaller projects showed that the design principle and the implemented modules work as intended.

6.5.2 Anticipations vs. Achievements

The low level expectation was a system for texture classification. The work resulted in a complex modular self-optimising system accompanied by a new texture definition and collection.

Having a re-usable system was the mid-level expectation. The resulting system is by intention biased towards texture classification but, as further projects showed, can easily be adopted for other tasks.

The final high-level expectation was to obtain ideas that lead towards artificial vision. The Texture Classification System can be regarded to bring knowledge a step closer to a real artificial vision system. Especially the complex yet flexible design, the interchangeability of modules and the self-optimisation routines show how modern computer vision systems have to be designed.

6.6 Positive Outcome of the Work

All summed up a new system has been created which has the capability to address the more demanding image processing tasks. Here, too, the sum is more than the single

elements and therefore some of the novelties designed or implemented in this work may not shine so bright as they should. One positive and useful design feature of the system is the possibility to include new ideas without the need for a system re-design.

6.7 Contribution to Knowledge

What is new? If only a single item shall be mentioned it has to be the Texture Classification System itself. But this is only a conglomeration of design principles, existing and new algorithms and new definitions.

Most important is the complex, flexible, self-optimising system design. Next to this a number of new ideas or approaches are described in this work. These find their way into the system as modules.

Important is also the way of how to build a self-optimising system. The use and usefulness of this approach has been discussed in detail.

Last but not least a new texture image library has been created and a new, more detailed definition of texture has been given.

6.8 Future Work

Artificial vision is the future. It will be needed for almost every area of science, industry and society. Even though this might sound like science fiction today, many computerised systems will be able to see, to perceive their surroundings, within the next twenty years. This estimation becomes more believable when one looks back two decades. Who in the early 1980s owned a computer, had a mobile phone or a car that tells you which route to take?[1] The development of computer hardware is very fast which leaves no doubt that the needed performance for artificial vision will be available in the estimated time.

But what is it good for? Artificial vision systems will be used in every area of quality control to driving a car for which today humans are needed.

The design of the system described in this work is the first instance of an artificial vision system.

[1] Today mobile phones are available that are hand held computers, take pictures *and* navigate your car.

With the principles at hand the work that has to be done over the following years and decades is quite clear. The system design and especially its complexity has to be used with increasingly faster hardware. Quantum effect computers, still in their infancy today, will provide suitable platforms in 10 to 15 years. The results obtained from such set-ups will show which modules, i. e. pre-processing, classification, etc., are most promising and will lead to a better understanding of the system's abilities and of vision's necessities. As vision is not texture alone new ideas will be included as new modules and tested against and combined with older ones. There will be different and even divergent results for specialised tasks, but in the end a universal artificial vision system will be the result.

Appendix A

Images of Employed Textures

The first images that were used for the system test were from the book of PHIL BRODATZ *Texture: A Photographic Album for Artists and Designers* (cf. Section A.1) and from the *Vision Texture Database* at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY MEDIA LAB (cf. Section A.2).

These textures are fairly easily distinguishable from one another and could only be regarded as a first trial set of textures. Therefore the author has started compiling his own texture image collection (cf. Section A.3).

Additionally some pictures from real world project related to texture classification will be displayed (cf. Sections B.1 - B.4).

A.1 The BRODATZ Texture Collection

Quite famous and until today one of the standard texture image collections are the images from PHIL BRODATZ [Bro66]. The original book contained 112 images, most of which were taken in a studio. The image in Fig. 108 shows the set-up used for taking the photographies. An 8×10 inch View Camera was used, for outdoor pictures a 4×5 inch Speed Graphic (Fig. 109) or a 5×7 inch View Camera. These were plate cameras that produced high quality images. The detail resolution of the original images is much high than that of the digitised images. The book as well as the single copies, which were obtainable from the photographer (at $\$25$ each) was intended, as the title of the book suggests, for artists and designers. In 1966 no one thought about texture classification on computers.

Figure 108: Studio set-up for the images by PHIL BRODATZ.

Thirteen out of 112 images were digitised with a resolution of 512 × 512 pixel and made available over the internet [USC]. Who did the digitisation and which equipment was used is not known. By comparing the digitised images to the prints in the original book, it is clear that single prints must have been used for digitisation. For example the image in Fig. 111.9 extends on two sides over the image 'D 68 Wood grain' in the book and Fig. 111.3 could not be matched to the book's image 'D 15 Straw' at all. In all images only a part of the original was digitised. The scans with the higher resolution (1024 × 1024 pixel) are not identical in scale and sector. Additionally the contrast is much lower compared to the prints. Figs. 111.1 to 111.13 show the available texture images.

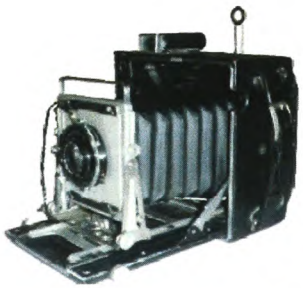
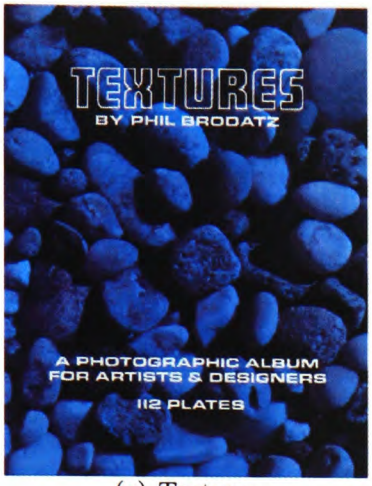
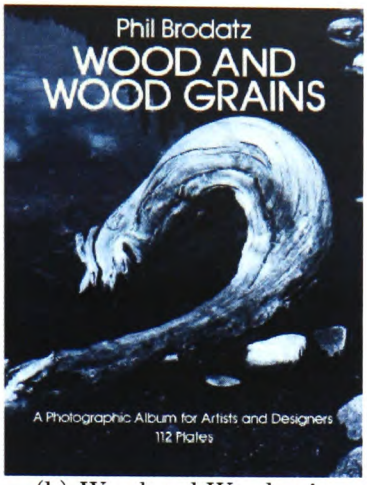


Figure 109: Speed Graphic camera used for outdoor images by PHIL BRODATZ.

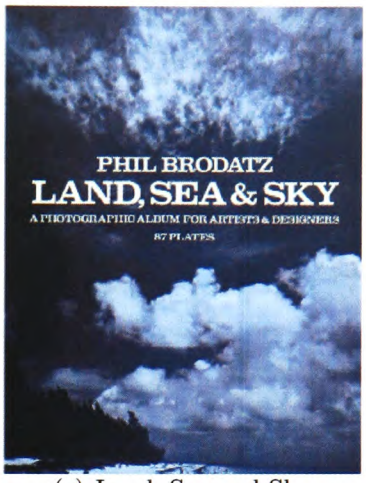
Even the legality of using the images for research has been doubted. PHIL BRODATZ only allowed using up to three images without charge when he published his book.



(a) Textures



(b) Wood and Woodgrain



(c) Land, Sea and Sky

Figure 110: Publications by PHIL BRODATZ containing texture images.

Completely unknown seems to be the fact that the next books by PHIL BRODATZ *Wood and Woodgrain* [Bro71] and *Land, Sea and Sky* [Bro76] also contained interesting texture images.

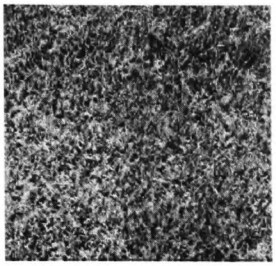
It appears that the BRODATZ Texture Collection only came into life because some fellow researcher happened to own the book[1] and bought the prints for digitisation. Ever since these images were used by many other researchers as they were easily obtainable and results could be compared.

The problem with the BRODATZ textures is that the originals are too good. From the photographic point of view they are excellently made, which means they are better than real life. But as the Texture Classification System has been designed for real life, other textures have to be used, too.

The specifications of the available images are:

- ▷ 512 × 512 pixel (262.144 pixel) or 1024 × 1024 pixel (1.048.576 pixel)
- ▷ 8-bit greylevel (256 different values)
- ▷ non-modified (the images have not been preprocessed, e. g. histogram equalised)

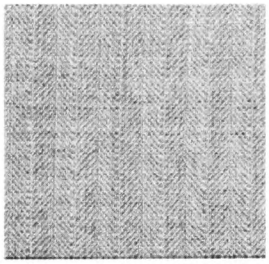
[1] Just like the LENNA picture used for image compression testing.



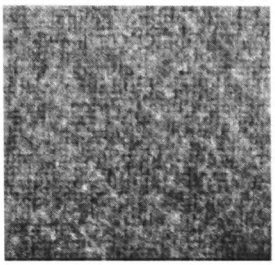
(111.1) D9

(111.2) D12

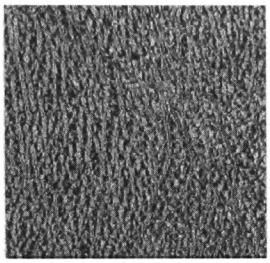
(111.3) D15



(111.4) D17

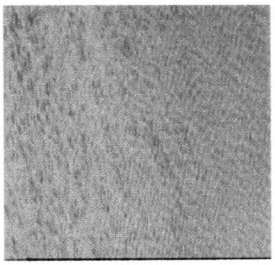


(111.5) D19

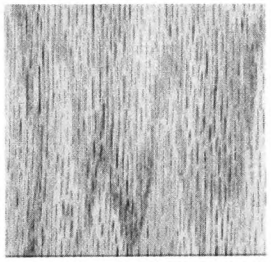


(111.6) D24

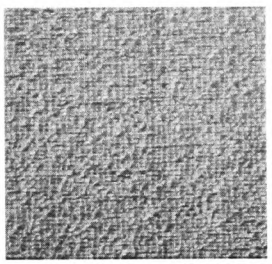
(111.7) D29



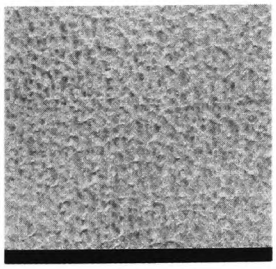
(111.8) D38



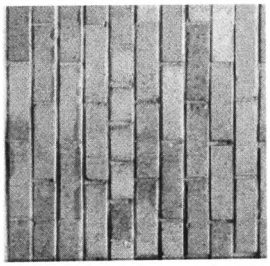
(111.9) D68



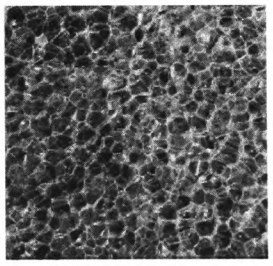
(111.10) D84



(111.11) D92



(111.12) D94



(111.13) D112

Figure 111: BRODATZ Textures

A.2 The VISTEX Texture Collection

The VISTEX[2] database is a collection of texture images[3] which was created in the mid-1990s with the intention of providing textures for computer vision applications and an alternative to the BRODATZ textures.

The images in the VISTEX collection do not all conform to rigid frontal plane perspectives and studio lighting conditions. The goal of VISTEX was to provide texture images that are representative of real world conditions. While the VISTEX collection can serve as a replacement for traditional texture collections, it includes examples of many non-traditional textures.

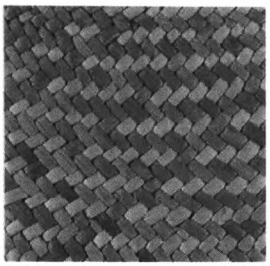
As of December 2002 the VISTEX database [Vis] is no longer maintained, but is only available "as is." Images with frontal plane perspective can be seen in Figs. 112.1 to 112.19.

[2]short for Vision Texture

[3]

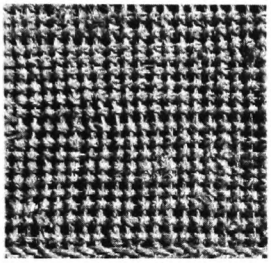

(112.1)

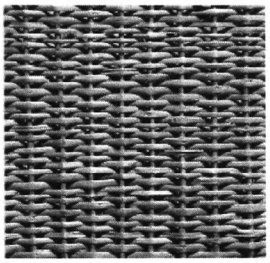

(112.2)

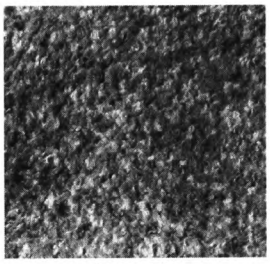

(112.3)

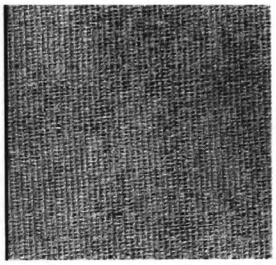

(112.4)

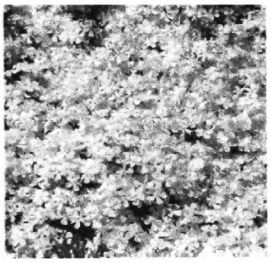

(112.5)


(112.6)


(112.7)


(112.8)


(112.9)


(112.10)

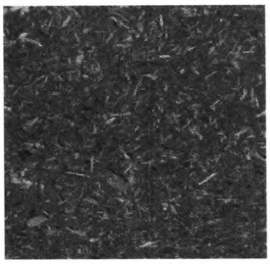

(112.11)


(112.12)

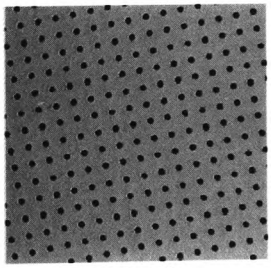

(112.13)


(112.14)


(112.15)


(112.16)


(112.17)


(112.18)

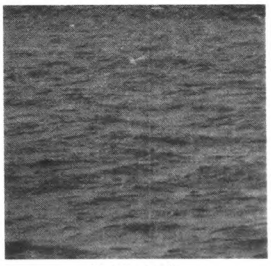

(112.19)

Figure 112: VISTEX Textures

A.3 The STOLPMANN Texture Collection

The limited number of available textures, copyright or maintenance problems initiated the author's own texture collection. At the beginning pictures were taken with an SLR camera on transparency film. The resulting slides were professionally printed onto picture paper and afterwards digitised using a flat-bed scanner. Later digitising was done directly using a slide-scanner with a resolution of up to 2400 dpi. This resulted in very large images of more than 8 Mpixels and 25 MByte in `.tiff` format. For further investigation they had to be reduced in size. The first images for the steel block project described in Section 4.4 were obtained in that way. Some images can be seen in Section B.3.

As every processing step introduces extra noise to an image, digital SLR cameras were used next. The first digital camera, which was used for the veneer project described in Section 4.3 was an Olympus camera. The image resolution is 1280×1024 pixel. It has to be mentioned, that the images could only be captured in `.jpg` format, which of course is not optimal. Some images can be seen in Section B.2.

Next a Canon D30 digital camera with an image resolution of 2160×1440 pixel was used (Fig. 113). Some of the images taken with that camera can be seen in Figs. 114.1 to 114.60.

Figure 113: Canon D30 digital camera used for texture images shown in Fig. 114.

Not all digitised images will be shown in this book, but only a selection, which gives a good overview. New texture images are being taken all the time, and some older ones on slides have not been digitised, yet.

The images of the STOLPMANN texture collection will be made available to the research community at the end of the research project. The textures will be pre-classified according to the author's system as described in Section 2.3.2.

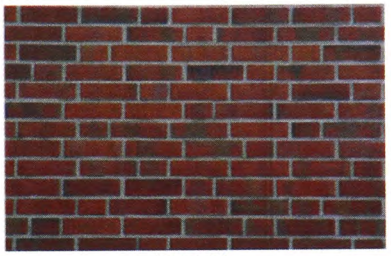

(114.1)

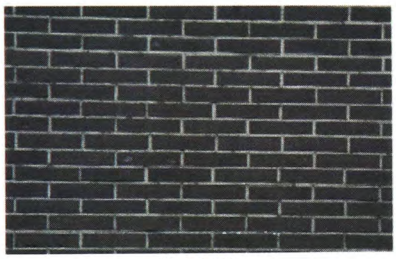

(114.2)

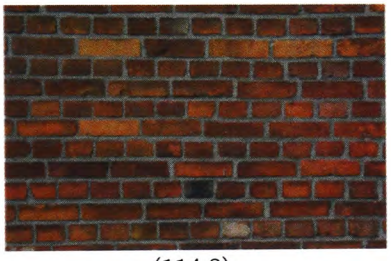

(114.3)


(114.4)


(114.5)

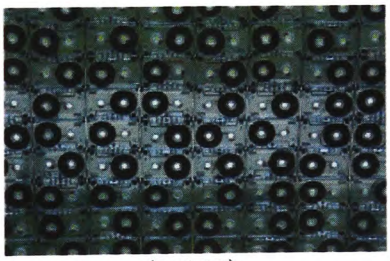

(114.6)

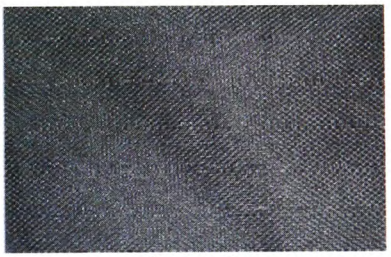

(114.7)


(114.8)


(114.9)

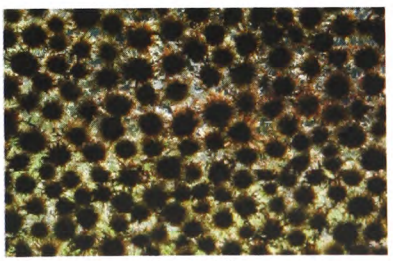

(114.10)


(114.11)

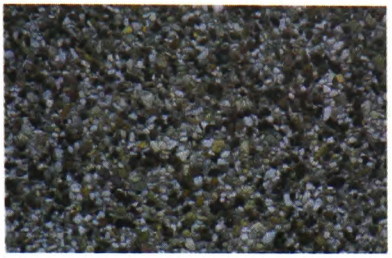

(114.12)


(114.13)

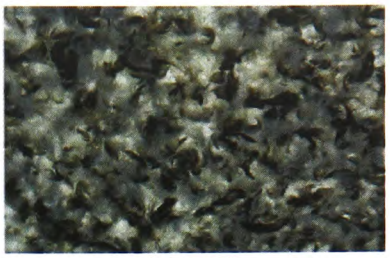

(114.14)


(114.15)

(114.16)

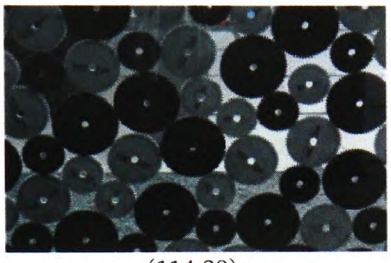


(114.17)



(114.18)

(114.19)



(114.20)

(114.21)

(114.22)

(114.23)

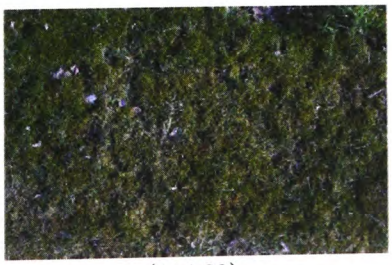
(114.24)

(114.25)

(114.26)

(114.27)

(114.28)



(114.29)

(114.30)


(114.31)

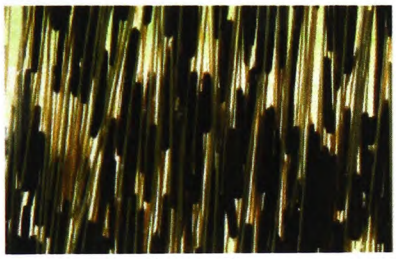

(114.32)


(114.33)


(114.34)


(114.35)


(114.36)


(114.37)


(114.38)


(114.39)


(114.40)

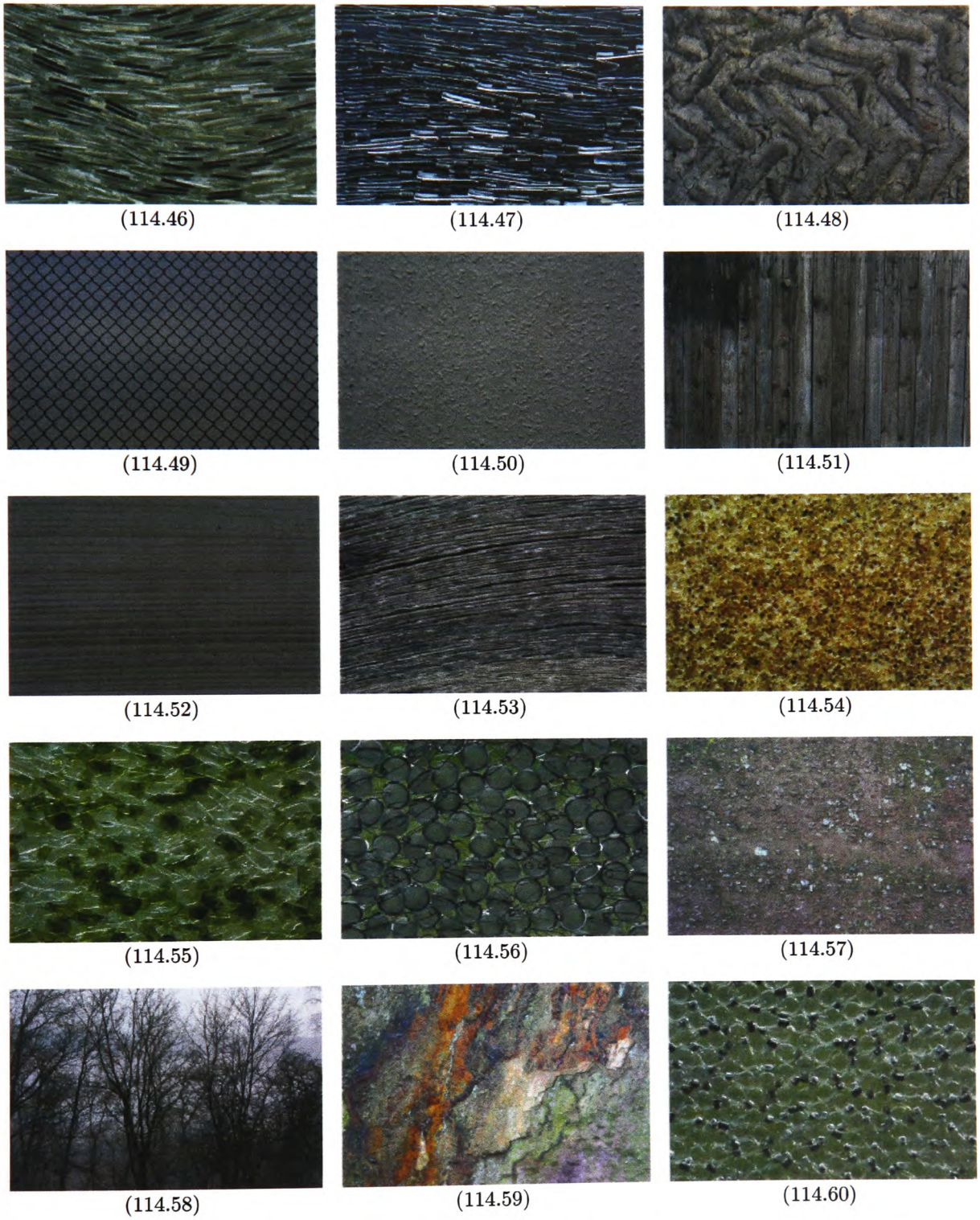

(114.41)


(114.42)


(114.43)


(114.44)


(114.45)



(114.46) (114.47) (114.48)

(114.49) (114.50) (114.51)

(114.52) (114.53) (114.54)

(114.55) (114.56) (114.57)

(114.58) (114.59) (114.60)

Figure 114: Some of the STOLPMANN Textures

Appendix B

Images from Projects

B.1 Image of the Glass Project

In this Section some more of the images used for the glass project described in Section 4.2 are displayed.

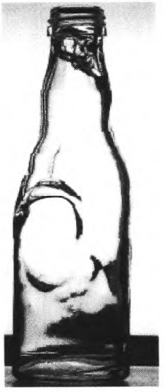
(115.1)

(115.2)

(115.3)

(115.4)

(115.5)



(115.6)

(115.7) (115.8) (115.9) (115.10) (115.11) (115.12)

(115.13) (115.14) (115.15) (115.16) (115.17) (115.18)

(115.19) (115.20) (115.21) (115.22) (115.23) (115.24)

Figure 115: Glass bottles with production faults.

B.2 Image of the Veneer Project

This section shows typical faults found in veneer. More on that project can be found in Section 4.3.

(116.1)

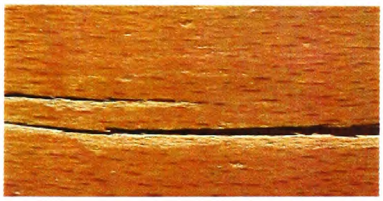
(116.2)

(116.3)

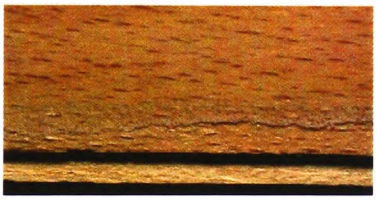
(116.4)

(116.5)

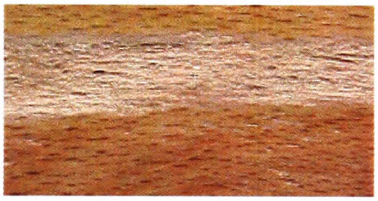
(116.6)



(116.7)

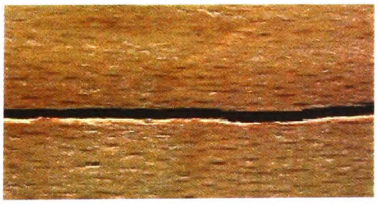


(116.8)

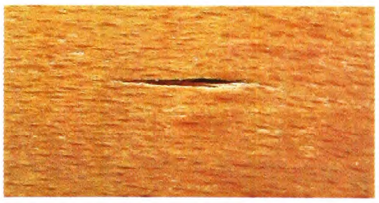


(116.9)

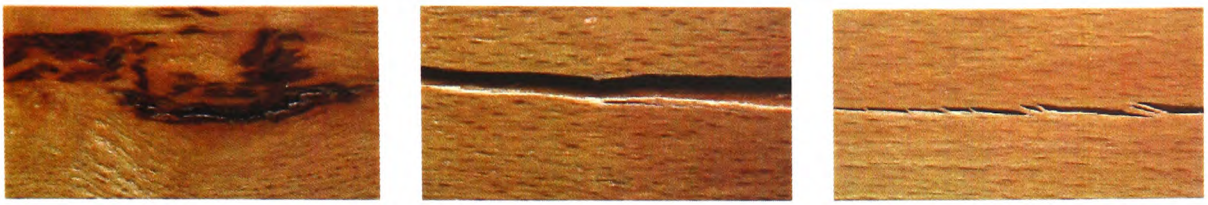
(116.10)



(116.11)



(116.12)

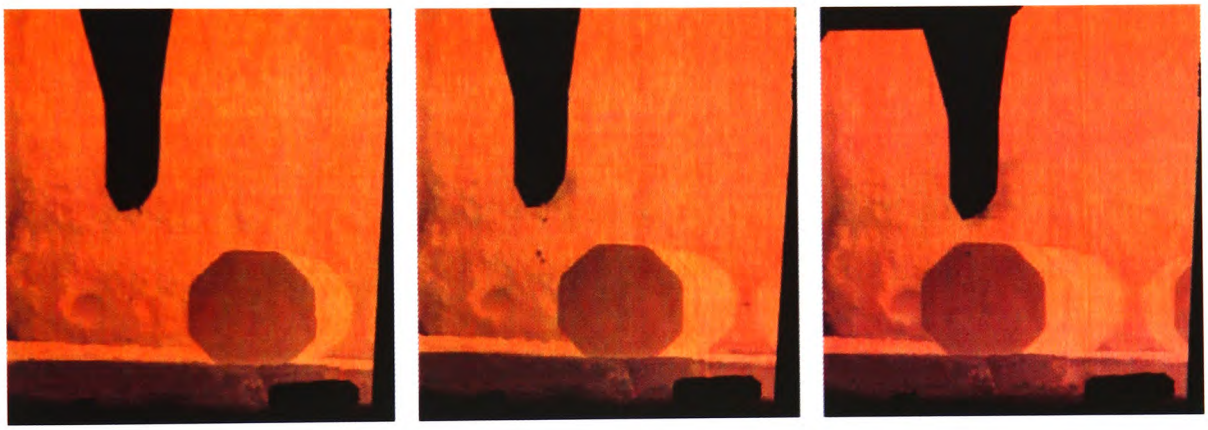


(116.13) (116.14) (116.15)

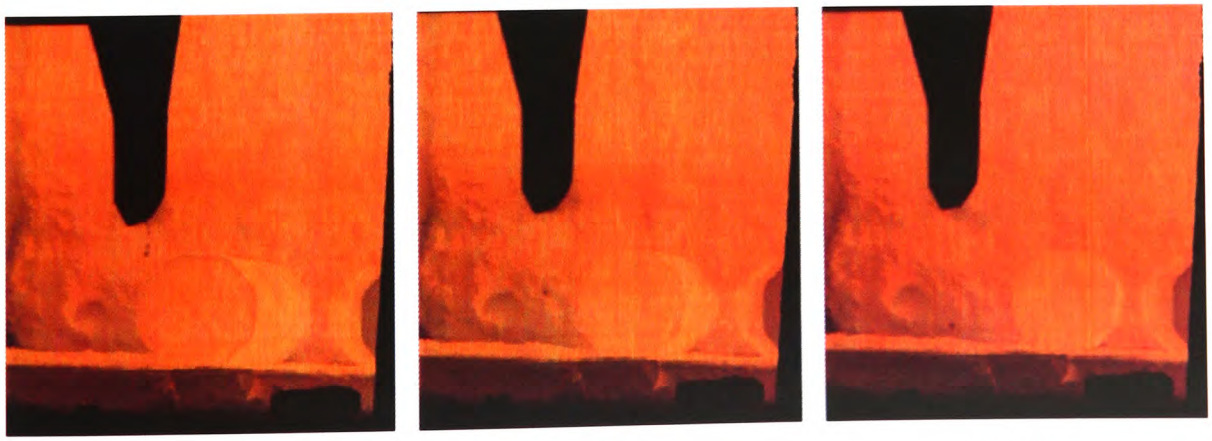
Figure 116: Assortment of faults in veneer.

B.3 Image of the Steel Project

The following images were made for the steel block project, described in Section 4.4.



(117.1) (117.2) (117.3)



(117.4) (117.5) (117.6)

(117.7)

(117.8)

(117.9)

(117.10)

(117.11)

(117.12)

(117.13)

(117.14)

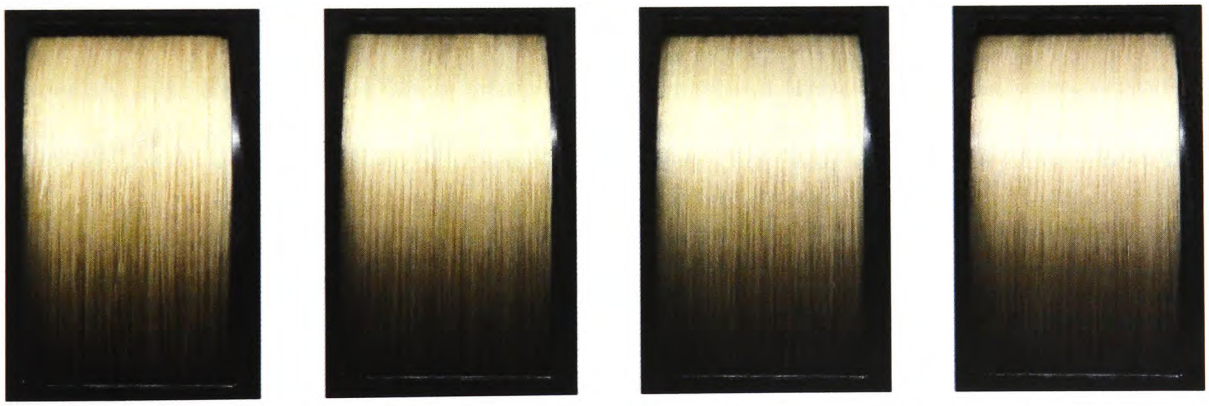
(117.15)

(117.16) (117.17) (117.18)

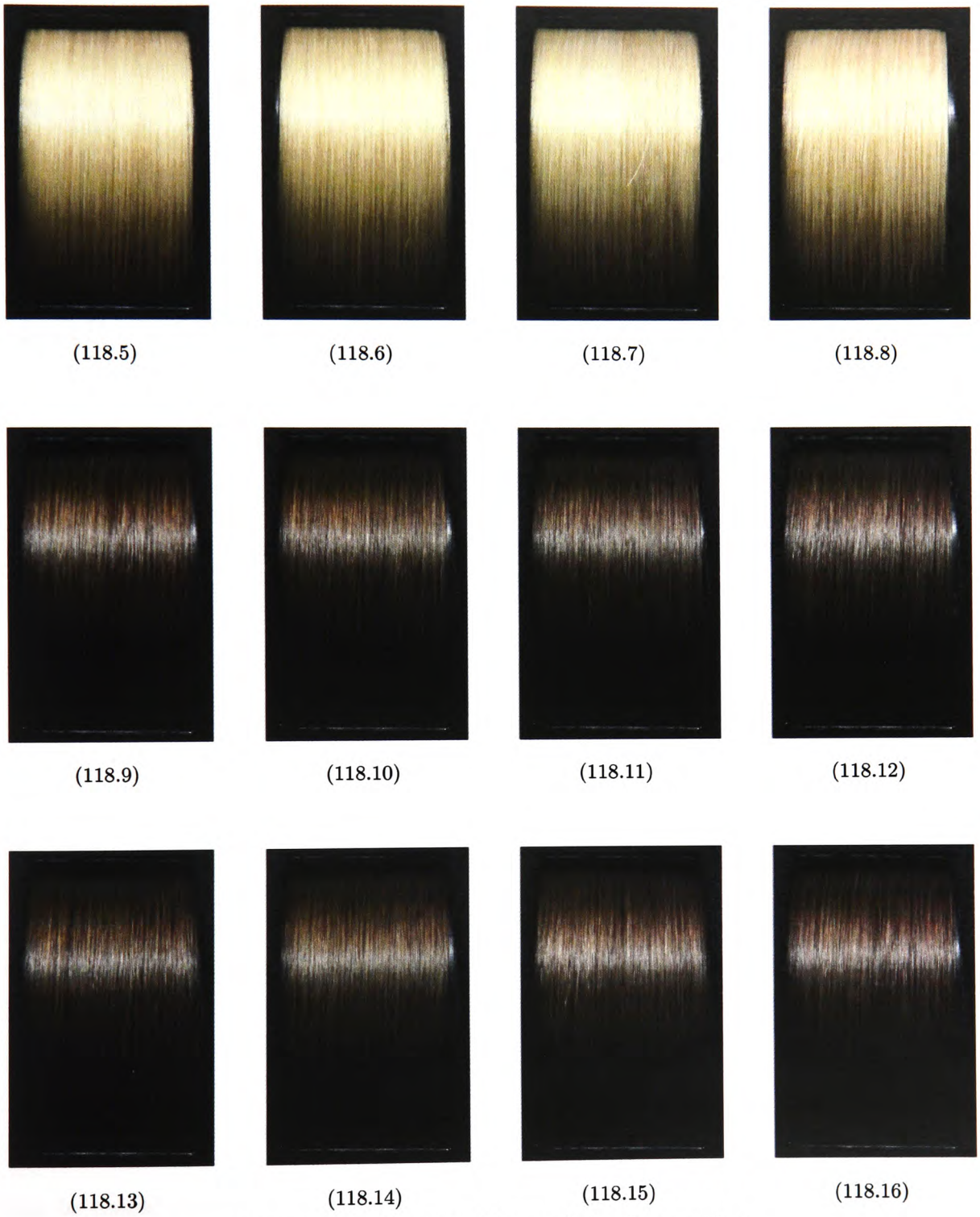
Figure 117: Steel blocks in a rotating oven.

B.4 Image of the Hair-Gloss Project

This Section shows some of the hair-probes of the hair-gloss project described in Section 4.5.



(118.1) (118.2) (118.3) (118.4)



(118.5) (118.6) (118.7) (118.8)

(118.9) (118.10) (118.11) (118.12)

(118.13) (118.14) (118.15) (118.16)

Figure 118: Blond and brown hair-gloss probes.

Appendix C

The WiT Image Processing Tool

C.1 General Introduction into the Programme

WiT is a powerful visual programming tool for designing computer algorithms with block diagrams, called iGraphs, which can be executed. iGraphs are built using icons and links with point-and-click simplicity. They are particularly suitable for complex image processing applications in areas such as machine vision, biomedical imaging, and imaging research.

C.1.1 WiT iGraphs

An iGraphs presents an entire algorithm using proven building blocks (icons) to perform image processing. Links direct data from one icon to the next and can be configured to enable probes or breakpoints. Links can be split to deliver the same data to many blocks. Parameters for operators are easily changed with quick pop-up dialogue panels. All the standard features of a CAD-like environment such as cut, copy, paste, undo, zoom, area select, grid snapping, and pan are supplied for easy iGraph design. The Fig. 119 shows an example iGraph and a typical WiT session.

Flow control is handled by special operators to perform looping, if-then-else, sequencing of large data sets, collecting, I/O, counting, and general data manipulation. iGraphs can be nested by creating operators which encapsulate an entire iGraph for use in other iGraphs. This hides unnecessary detail and promotes reuse of common processing steps.

There are hundreds of processing functions that comes standard with WiT. To add new functions a C-code algorithm is used to create a new operator. This then can be used iconically as efficiently as any built-in WiT operator.

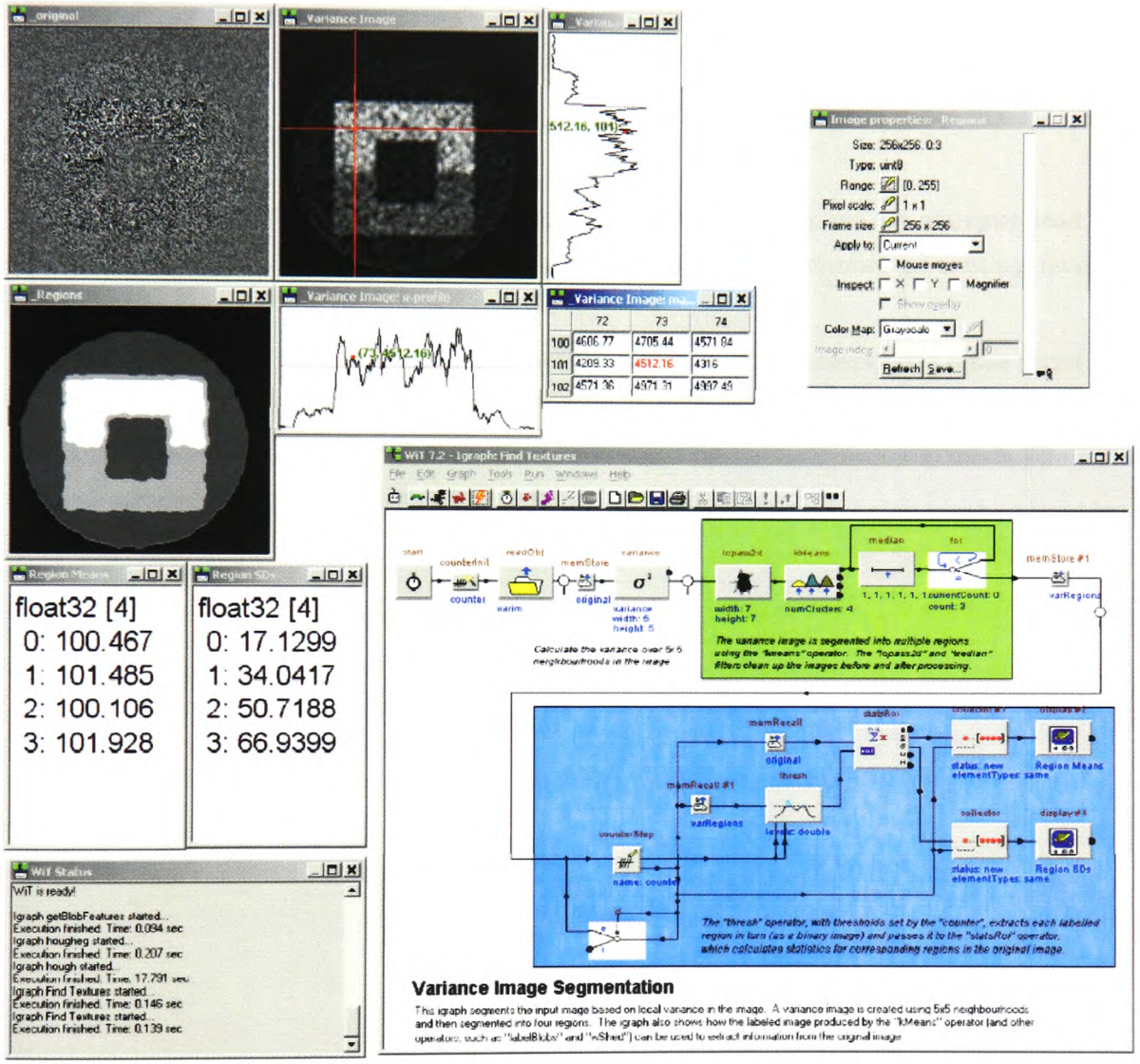


Figure 119: A WiT example iGraph.

iGraphs are like block diagrams. They are much more intuitive and less restrictive than traditional linear textual programming code. Because iGraphs use links to connect data paths between operators, variable names are not required. This makes experimental

development of new algorithms much easier because operators and execution sequences can be changed without having to rename data variables.

An iGraph is a graphical representation of an imaging algorithm. On an iGraph, each icon represents an operation, such as reading an image file, performing a convolution, or generating a lookup table. Operators are represented by graphical icons.

C.1.1.1 Sub-iGraphs

When an iGraph becomes too complex, portions of the iGraph can be encapsulated into a sub-iGraph to clarify the design. There is no limit to the number of nesting levels of sub-iGraphs.

The sub-iGraph contains input and output ports which corresponds to the input and output ports on the sub-iGraph icon.

Sub-iGraphs can also be used as a way to re-use design components, much like functions or procedures in a traditional programming language.

Appendix D

Neural Network Software

D.1 The Stuttgarter Neuronale Netze Simulator

The Stuttgarter Neuronale Netze Simulator (SNNS) is a software simulator for neural networks on Unix workstations developed at the Institute for Parallel and Distributed High Performance Systems (IPVR) at the University of Stuttgart. The goal of the SNNS project is to create an efficient and flexible simulation environment for research on and application of neural networks.

The SNNS simulator consists of four main components (cf. Fig. D.1):

1. the simulator kernel written in C,
2. the graphical user interface XGUI under X11R4 or X11R5,
3. the scripting link *batchman*,
4. and the network compiler *SNNS 2 C*.

The simulator kernel operates on the internal network data structures of the neural nets and performs all operations of learning and recall. It can also be used without the other parts as a C program embedded in custom applications. SNNS can be extended by the user with user defined activation functions, output functions, site functions and learning procedures, which are written as simple C programs and linked to the simulator kernel.

According to [SNN] currently the following network architectures and learning procedures are included in the latest version `SNNS 4.2`:

- ▷ Backpropagation (BP) for feedforward networks
- ▷ Counterpropagation
- ▷ Quickprop
- ▷ Backpercolation 1
- ▷ RProp
- ▷ Generalised radial basis functions (RBF)
- ▷ ART1
- ▷ ART2
- ▷ ARTMAP
- ▷ Cascade Correlation
- ▷ Recurrent Cascade Correlation
- ▷ Dynamic LVQ
- ▷ Backpropagation through time (for recurrent networks)
- ▷ Quickprop through time (for recurrent networks)
- ▷ Self-orgamising maps (Kohonen maps)
- ▷ TDNN (time-delay networks) with Backpropagation
- ▷ Jordan networks
- ▷ Elman networks and extended hierarchical Elman networks
- ▷ Associative Memory

The X Graphical User Interface (X GUI), built on top of the kernel, gives a 2D and a 3D graphical representation of the neural networks and controls the kernel during the simulation run. In addition, the 2D user interface has an integrated network editor which can be used to directly create, manipulate and visualise neural nets in various ways.

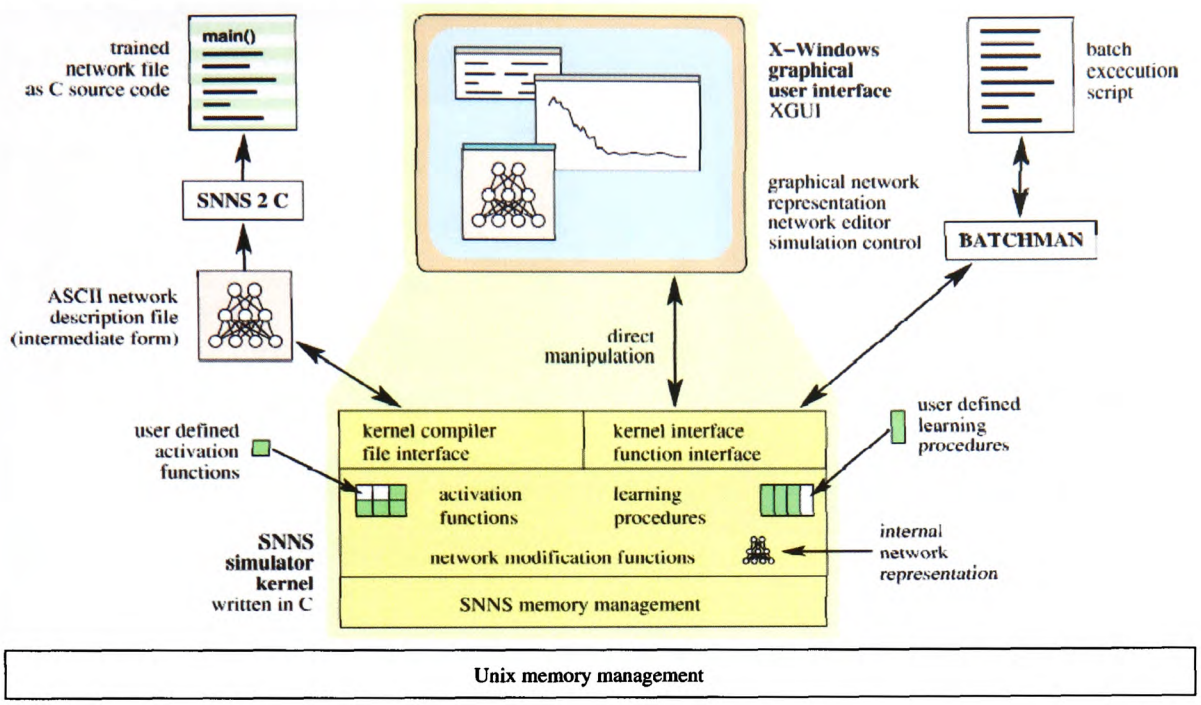


Figure 120: The SNNS software system layout. (adapted from [SNN])

The *batchman* is the pendant to X GUI. If no graphical interface is needed, experienced users may use the scripting form instead.

The programme *SNNS 2 C* can, with certain limitations, convert a trained net into C-code. This code can then be used in other environments, e. g. in WiT.

To be able to use SNNS under a WINDOWS operating system an X-Server[1] is needed.

D.1.1 The JAVA NEURONALE NETZE SIMULATOR

The Java Neuronale Netze Simulator (JavaNNS) is the successor of SNNS. It is based on its computing kernel, with a newly developed graphical user interface written in Java set on top of it. Hence the compatibility with SNNS is achieved, while the platform-independence is increased.

In addition to SNNS features, JavaNNS offers the possibility of linking HTML browsers to it which makes it possible to call for the user manual, which is available in HTML format, from within the program.

[1] X is the windowing system in the UNIX world.

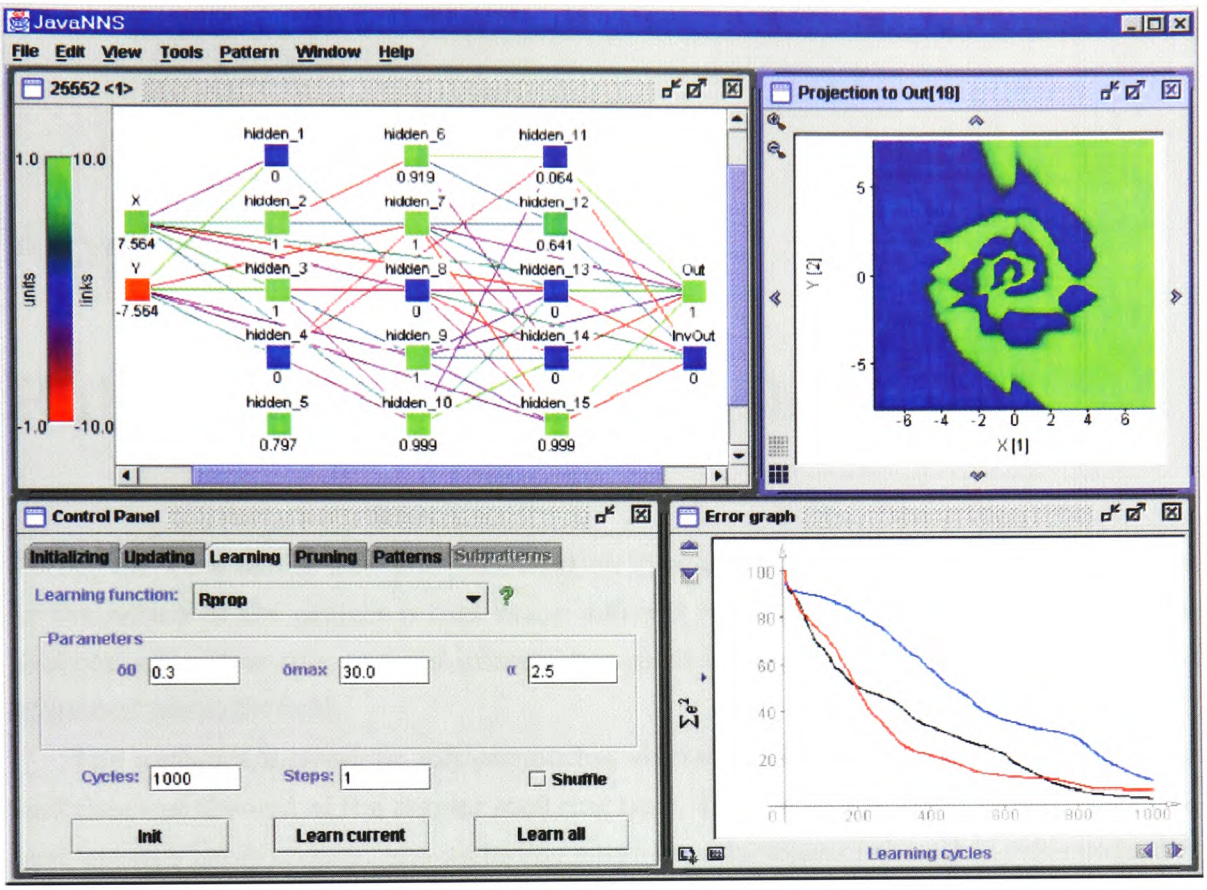


Figure 121: The JavaNNS graphical user interface. [Jav]

According to [Jav], JavaNNS version 1.1 is available for WINDOWS (NT 4 / 2000), SOLARIS and REDHAT LINUX.

D.1.2 WiT and the SNNS Software

At the moment a number of *Diplomarbeiten* (final year thesis') are being written exploiting the possibilities of the SNNS software. Especially the use of SNNS and JavaNNS in combination with WiT is being described in [Hei03] and [Sen04].

Tools have been written for the conversion of preprocessed input and classified output data to and from SNNS.

Appendix E

Publications of the Author

During the work on the Texture Classification System a number of papers were published. As the nature of the project is that many different research areas are being included, the publications mirror this. A brief introduction shall be given, describing the contents and addressed scientific field.

The author's interest in soft-computing started with his *Diplomarbeit* [Sto93], the final thesis at the end of the regular studying time. It is an introduction to fuzzy logic, biased towards fuzzy control, and including short introductions of the other soft-computing areas.

The interest in soft-computing and in image processing was the basis for the Texture Classification System. An overview over the goals of the system was given in the *Transfer Document* [Sto97]. This was followed by a number of international conferences to assess the response of fellow researchers.

The first was the ICSP in Beijing, China. Signal processing is the main subject of this annual conference. Being one of the larger conferences many image processing researchers attended. A paper about the use of genetic algorithms for optimisation was presented. [SD98]

The NAFIPS conference in New York, USA, has a long tradition in bringing people together working on fuzzy logic. Fuzzy clustering for texture classification was the subject of the paper presented there. [SD99a]

Oulu in Finland was the place for the TEXTURE workshop bringing together the comparably small community of texture researchers. The paper presented there was on high-level statistics and soft-computing. [SD99b]

At the EUFIT conference in Aachen, Germany, an annual event for fuzzy research, a paper about the feature vector optimisation was presented. [SD99c]

A paper about the project on quality control in veneer was presented at the annual North German Fuzzy Symposium in Clausthal-Zellerfeld, Germany. A longer version was later published in the AFN-Book. [SAD00b, SAD00c]

A paper on statistics and soft-computing methods for texture classification was published in the *Texture Analysis Book*. [SD00]

An application of texture analysis, namely veneer, was published in the book on *Evolutionary Design and Manufacture* which followed a presentation at the ACDM conference in Plymouth, UK. It focused on soft-computing methods. [SAD00a]

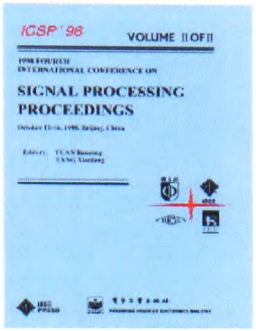
A number of reports and shorter publications were written in connection with the project on quality control of glass containers (cf. Section 4.2). Next to an oral presentation at the North German Fuzzy Symposium the project was displayed at the Hanover Industrial Fair in 2001. This project utilised the concepts and ideas as well as modules from the Texture Classification System. [SA01a, SA01b, SA01c]

Some more papers and reports discussed the possibilities of reading badly printed or smeared text. [SS03, SS04a, SS04b, Sto04] A number of master theses were written about this subject under the author's guidance, too. [Lóp03, Hei02, Hei03, GT03, Sen04] The principles of the Texture Classification System were again used. The work of [Lóp03] was used to produce two different systems able to read and classify text displayed on Colour-LCD-Screens. Those systems are used at BOSCH/BLAUPUNKT to test the software of car navigation and car radio devices. The author's contribution and the work of [Hei03, GT03, Sen04] was presented at the Hanover Industrial Fair in 2004. [Sto04] was presented at 7th Göttinger Symposium on Soft Computing.

The very latest projects that benefit from the author's research are driver-assistance-systems. The subject itself is still in its infancy but highly regarded in the automotive industry. Once the computer hardware is fast enough such systems will be able to take over the control of a vehicle. Until then more and more systems will be introduced that assist and guide the driver. Two master theses were finished under the author's supervision in 2004 [Fel04, Mat04] and some more are in the pipeline. Recently a system based on those ideas was produced that guides a pole-setting robot along the street curb.

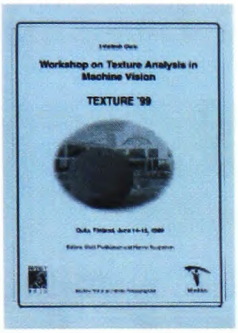
(122.1) [Sto93]

(122.2) [Sto97]



(122.3) [SD98]

(122.4) [SD99a]



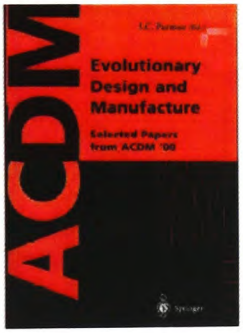
(122.5) [SD99b]

(122.6) [SD99c]

(122.7) [SAD00b]

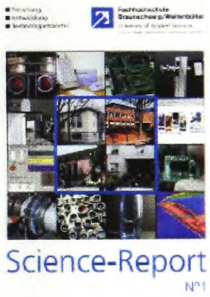
(122.8) [SAD00c]

(122.9) [SD00]



(122.10) [SAD00a]

(122.11) [SA01b]



(122.12) [SA01a]

(122.13) [SS04a]

(122.14) [Sto04]

Figure 122: Publications.

Glossary

Explanation of some of the technical and scientific terms used in the book.

Adaptive Thresholding

A threshold operation performed changing the threshold value based on local brightness characteristics of an image.

Additive Primaries

Red, Green and Blue light. All three added together gives white light.

Affine Warp

A first order spatial warping transformation, for example translation, rotation, or scaling.

Aliasing

Undesirable data anomalies present in image data. See also spatial and temporal aliasing.

Analogue to Digital Converter (ADC)

Element that converts analogue input to digital format.

Arithmetic and Logic Unit (ALU)

An element that performs linear and nonlinear image processing operations. It is usually part of the computer's main processor.

Artefact

An undesirable erroneous pixel value in an image.

Aspect Ratio

The ratio of an image's width to its height.

Bilinear Interpolation

Process which extracts the best value for a result pixel based on a weighted average of the four pixels surrounding the corresponding source pixel.

Binarisation

Process of transforming an image such that each pixel is mapped to one of two values based on a threshold value. For example, any pixel with an intensity above the threshold value is mapped to white and any pixel below the threshold value is mapped to black, or vice versa.

Binary Image

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or 255 for white.

Binary images are often produced by applying a threshold to a greyscale or colour image, in order to separate an object in the image from the background. The colour of the object (usually white) is referred to as the foreground colour. The rest (usually black) is referred to as the background colour. However, depending on the image to which a threshold is to be applied, this polarity might be inverted, in which case the object is displayed with 0 and the background is with a non-zero value.

Some morphological operators assume a certain polarity of the binary input image so that if an image is processed with inverse polarity the operator will have the opposite effect.

Bitmap

A type of graphic format in which the image is made up of a large number of dots arranged on a closely spaced grid.

Bits

"Binary Digits" (zero and 1). Smallest pieces of data recognisable to a computer.

Blob Analysis

The analysis of an image's objects, for example the number of objects, the size and orientation of the objects, etc. A blob contains a collection of contiguous pixels of the same or similar intensities, usually belonging to the same object.

Blob Labelling

The process of identifying the blobs in an image. A 4- or 8-neighbourhood may be used.

Blur Filter

A filter produced by replacing each pixel in an image with the average value of all of the pixels in its neighbourhood. More commonly called low-pass filter.

Boxcar Filter

A filter created by subtracting a low-pass filtered image from the original image. Accentuates the high frequency areas in an image. More commonly called high-pass filter.

Brightness

The attribute of a colour explaining how dark or light the colour is.

Brightness Contouring

A condition caused by too little brightness resolution. Makes gradual brightness changes appear abrupt instead. For example, if the number of brightness levels is changed from 16 to 8, the image will appear as if it were a "paint by number" painting or poster.

Brightness Slicing

A double contrast enhancement which produces a binary image by assigning pixels with intensity values below a lower threshold and above a higher threshold a value of black and pixels with within the upper and lower threshold ranges a value of white.

Centroid

Also known as the centre of mass. This is the point in an object where there is equal mass to the left, right, above and below.

Chroma

The attribute of a colour that describes how saturated a colour is.

Closing

Binary or grey-scale morphological operation that is accomplished by performing a dilation operation followed by an erosion operation. This cleans up the

image by removing small holes and then returning the remaining objects to their original size.

Colour Model

A colour measurement scale or system that numerically specifies the perceived attributes of colour. Used in computer graphics applications and by colour measurement instruments as well as for describing the characteristics of a colour sensor.

Colour Image

It is possible to construct (almost) all visible colours by combining the three primary colours red, green and blue, because the human eye has only three different colour receptors, each of them sensible to one of the three colours. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately 7 million colour shades [Bös95]. A RGB colour image is a multi-spectral image with one band for each colour red, green and blue, thus producing a weighted combination of the three primary colours for each pixel.

A full 24-bit colour image contains one 8-bit value for each colour, thus being able to display $2^{24} = 16777216$ different colours.

However, it is computationally expensive and often not necessary to use the full 24-bit image to store the colour for each pixel. Therefore, the colour for each pixel is often encoded in a single byte, resulting in an 8-bit colour image. The process of reducing the colour representation from 24-bits to 8-bits, known as colour quantisation, restricts the number of possible colours to 256. However, there is normally no visible difference between a 24-colour image and the same image displayed with 8 bits. An 8-bit colour images are based on colourmaps, which are look-up tables taking the 8-bit pixel value as index and providing an output value for each colour.

Colour Quantisation

Colour quantisation is applied when the colour information of an image is to be reduced. The most common case is when a 24-bit colour image is transformed into an 8-bit colour image.

Two decisions have to be made:

1. which colours of the larger colour set remain in the new image, and
2. how are the discarded colours mapped to the remaining ones.

The simplest way to transform a 24-bit colour image into 8 bits is to assign 3 bits to red and green and 2 bits to blue (blue has only 2 bits, because of the eye's lower sensitivity to this colour). This quantises to 8 different shades of red and green and 4 of blue. However, this method can yield only poor results. For example, an image might contain different shades of blue which are all clustered around a certain value such that only one shade of blue is used in the 8-bit image and the remaining three blues are not used.

Alternatively, since 8-bit colour images are displayed using a colourmap, any arbitrary colour can be assigned to each of the 256 8-bit values and a separate colourmap can be defined for each image. This enables a colour quantisation adjusted to the data contained in the image. One common approach is the popularity algorithm, which creates a histogram of all colours and retains the 256 most frequent ones. Another approach, known as the median-cut algorithm, yields even better results but also needs more computation time. This technique recursively fits a box around all colours used in the RGB colourspace which it splits at the median value of its longest side. The algorithm stops after 255 recursions. All colours in one box are mapped to the centroid of this box.

All above techniques restrict the number of displayed colours to 256. A technique of achieving additional colours is to apply a variation of half-toning used for grey scale images, thus increasing the colour resolution at the cost of spatial resolution. The 256 values of the colourmap are divided into four sections containing 64 different values of red, green, blue and white. As can be seen in Fig. 123, a 2×2 pixel area is grouped together to represent one composite colour, each of the four pixels displays either one of the primary colours or white. In this way, the number of possible colours is increased from 256 to 64^4 ($= 2^{24}$).

Colour Space

A three-dimensional geometric representation of the colours that can be seen and/or generated using a certain colour model and quantitatively measured.

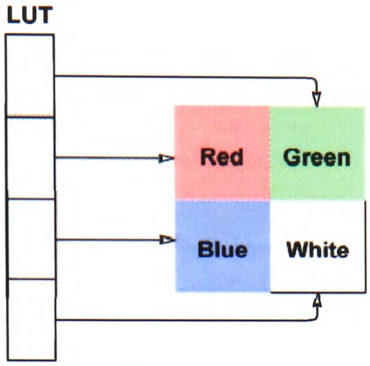


Figure 123: A 2×2 pixel area displaying one composite colour.

Colour Temperature

A measurement of the colour of light radiated by an object while it is being heated. Measured in Kelvin, lower temperatures are at 2400 K, and higher are at 9300 K.

Convolution

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution provides a way of 'multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.

In an image processing context, one of the input arrays is normally just a greylevel image. The second array is usually much smaller, and is also two-dimensional (although it may be just a single pixel thick), and is known as the kernel. Fig. 124 shows an example image and kernel that will be used to illustrate convolution. The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image. Each kernel position corresponds to a single output pixel, the value of which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together.

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

Figure 124: An example small image (left) and kernel (right) to illustrate convolution. The labels within each grid square are used to identify each square.

In the example, the value of the bottom right pixel in the output image will be given by:

$$O_{57} = I_{57}K11 + I_{58}K12 + I_{59}K13 + I_{67}K21 + I_{68}K22 + I_{69}K23 \quad (64)$$

If the image has M rows and N columns, and the kernel has m rows and n columns, then the size of the output image will have $M - m + 1$ rows, and $N - n + 1$ columns.[1]

Mathematically the convolution is:

$$O_{i,j} = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i + k - 1, j + l - 1)K(k, l) \quad (65)$$

where i runs from 1 to $M - m + 1$ and j runs from 1 to $N - n + 1$.

Convolution can be used to implement many different operators, particularly spatial filters and feature detectors. Examples include Gaussian smoothing and the Sobel edge detector.

[1]Note that many implementations of convolution produce a larger output image than this because they relax the constraint that the kernel can only be moved to positions where it fits entirely within the image. Instead, these implementations typically slide the kernel to all positions where just the top left corner of the kernel is within the image. Therefore the kernel 'overlaps' the image on the bottom and right edges. One advantage of this approach is that the output image is the same size as the input image. Unfortunately, in order to calculate the output pixel values for the bottom and right edges of the image, it is necessary to invent input pixel values for places where the kernel extends off the end of the image. Typically pixel values of zero are chosen for regions outside the true image, but this can often distort the output image at these places.

Correlation

A function used to compare an image with a small reference image of the object (pattern) to be matched. The application in image processing is known as template matching.

Differencing

Subtraction of one image from another. Used for motion detection and inspection to detect differences between image to be inspected and 'perfect' image.

Digitisation

The sampling of an analogue video signal at discrete intervals to create a digital image.

Dilation

Morphological process that looks at an image and for each neighbourhood, if any pixel in the neighborhood is 'active' the entire neighbourhood in the corresponding neighbourhood in the result image will be active.

Distance Metrics

It is often useful in image processing to be able to calculate the distance between two pixels in an image, but this is not as straightforward as it seems. The presence of the pixel grid makes several so-called distance metrics possible which often give different answers to each other for the distance between the same pair of points. The three most important ones are:

Euclidean Distance

This is the familiar straight line distance that most people are familiar with. If the two pixels that are considered have coordinates (x_1, y_1) and (x_2, y_2), then the Euclidean distance is given by:

$$D_{\text{Euclid}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{66}$$

City Block Distance

Also known as the Manhattan distance. This metric assumes that in going from one pixel to the other it is only possible to travel directly along pixel grid lines. Diagonal moves are not allowed. Therefore the 'city block' distance is given by:

$$D_{\text{City}} = |x_2 - x_1| + |y_2 - y_1| \tag{67}$$

Chessboard Distance

This metric assumes that you can make moves on the pixel grid as if you were a King making moves in chess, i. e. a diagonal move counts the same as a horizontal move. This means that the metric is given by:

$$D_{\text{Chess}} = \max(|x_2 - x_1|, |y_2 - y_1|) \tag{68}$$

Note that the last two metrics are usually much faster to compute than the Euclidean metric and so are sometimes used where speed is critical but accuracy is not too important.

Dithering

Dithering is an image display technique that is useful for overcoming limited display resources. The word dither refers to a random or semi-random perturbation of the pixel values.

Two applications of this technique are particularly useful:

- ▷ Low quantisation display: When images are quantised to a few bits (e. g. 3) then only a limited number of greylevels are used in the display of the image. If the scene is smoothly shaded, then the image display will generate rather distinct boundaries around the edges of image regions when the original scene intensity moves from one quantisation level to the next. To eliminate this effect, one dithering technique adds random noise (with a small range of values) to the input signal before quantisation into the output range. This randomises the quantisation of the pixels at the original quantisation boundary, and thus pixels make a more gradual transition from neighbourhoods containing 100% of the first quantisation level to neighbourhoods containing 100% of the second quantisation level.
- ▷ Limited colour display: When fewer colours are able to be displayed (e. g. 256) than are present in the input image (e. g. 24 bit colour), then patterns of adjacent pixels are used to simulate the appearance of the unrepresented colours.

Edge Detector

Edges are places in the image with strong intensity contrast. Since edges often occur at image locations representing object boundaries, edge detection is

extensively used in image segmentation when we want to divide the image into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while retaining most of the image information.

Since edges consist of mainly high frequencies, we can, in theory, detect edges by applying a highpass frequency filter in the Fourier domain or by convolving the image with an appropriate kernel in the spatial domain. In practice, edge detection is performed in the spatial domain, because it is computationally less expensive and often yields better results.

Since edges correspond to strong illumination gradients, we can highlight them by calculating the derivatives of the image. This is illustrated for the one-dimensional case in Fig. 125. We can see that the position of the edge can

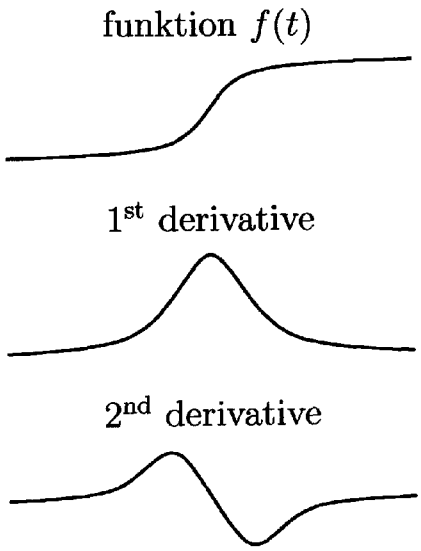


Figure 125: Sketch of 1st and 2nd derivative of an edge illustrated in one dimension.

be estimated with the maximum of the 1st derivative or with the zero-crossing of the 2nd derivative. Therefore we want to find a technique to calculate the derivative of a two-dimensional image. For a discrete one-dimensional function $f(i)$, the first derivative can be approximated by

$$\frac{df(i)}{d(i)} = f(i+1) - f(i) \tag{69}$$

Calculating this formula is equivalent to convolving the function with $[-11]$. Similarly the 2^{nd} derivative can be estimated by convolving $f(i)$ with $[1 - 21]$.

Different edge detection kernels which are based on the above formula enable us to calculate either the 1^{st} or the 2^{nd} derivative of a two-dimensional image. There are two common approaches to estimate the 1^{st} derivative in a two-dimensional image, Prewitt compass edge detection and gradient edge detection.

Prewitt compass edge detection involves convolving the image with a set of (usually 8) kernels, each of which is sensitive to a different edge orientation. The kernel producing the maximum response at a pixel location determines the edge magnitude and orientation. Different sets of kernels might be used: examples include Prewitt, Sobel, Kirsch and Robinson kernels.

Gradient edge detection is the second and more widely used technique. Here, the image is convolved with only two kernels, one estimating the gradient in the x-direction, Gx, the other the gradient in the y-direction, Gy. The absolute gradient magnitude is then given by

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{70}$$

and is often approximated with

$$|G| = |Gx| + |Gy| \tag{71}$$

In many implementations, the gradient magnitude is the only output of a gradient edge detector, however the edge orientation might be calculated with

$$\theta = \arctan(\frac{Gy}{Gx}) - \frac{3\pi}{4} \tag{72}$$

The most common kernels used for the gradient edge detector are the Sobel, Roberts Cross and Prewitt operators. After having calculated the magnitude of the 1^{st} derivative, we now have to identify those pixels corresponding to an edge. The easiest way is to threshold the gradient image, assuming that all pixels having a local gradient above the threshold must represent an edge. An alternative technique is to look for local maxima in the gradient image, thus producing one pixel wide edges. A more sophisticated technique is used by the Canny edge detector. It first applies a gradient edge detector to the image and then finds the edge pixels using non-maximal suppression and hysteresis tracking.

An operator based on the 2^{nd} derivative of an image is the Marr edge detector, also known as zero crossing detector. Here, the 2^{nd} derivative is calculated using a Laplacian of Gaussian (LoG) filter. The Laplacian has the advantage that it is an isotropic measure of the 2^{nd} derivative of an image, i. e. the edge magnitude is obtained independently from the edge orientation by convolving the image with only one kernel. The edge positions are then given by the zero-crossings in the LoG image. The scale of the edges which are to be detected can be controlled by changing the variance of the Gaussian.

A general problem for edge detection is its sensitivity to noise, the reason being that calculating the derivative in the spatial domain corresponds to accentuating high frequencies and hence magnifying noise. This problem is addressed in the Canny and Marr operators by convolving the image with a smoothing operator (Gaussian) before calculating the derivative.

Erosion

A morphological process in which for each pixel in the source image, if any pixel in that neighborhood is not active (i.e., has a zero value), the corresponding result pixel in the target image will not be active. This process is used to clean up small bright spots or spikes in a binary image.

FIR (Finite Impulse Response) Filter

Another term for convolution.

Fourier Transform

Frequency space transformation that decomposes a spatial image into a set of sinusoidal frequency components.

Frame Grabber

A hardware device that is capable of acquiring, storing and displaying digital images.

Frequency Domain

For simplicity, assume that the image I being considered is formed by projection from scene S (which might be a two- or three-dimensional scene).

The frequency domain is a space in which each image value at image position F represents the amount that the intensity values in image I vary over a specific

distance related to F. In the frequency domain, changes in image position correspond to changes in the spatial frequency, or the rate at which image intensity values are changing in the spatial domain image I.

The spatial frequency domain is interesting because it may make explicit periodic relationships in the spatial domain, and some image processing operators are more efficient or indeed only practical when applied in the frequency domain. In most cases, the Fourier Transform is used to convert images from the spatial domain into the frequency domain and vice-versa.

A related term used in this context is spatial frequency, which refers to the (inverse of the) periodicity with which the image intensity values change. Image features with high spatial frequency (such as edges) are those that change greatly in intensity over short image distances.

Greyscale Image

A greyscale (or greylevel) image is simply one in which the only colours are shades of grey. The reason for differentiating such images from any other sort of colour image is that less information needs to be provided for each pixel. In fact a 'grey' colour is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full colour image.

Often, the greyscale intensity is stored as an 8-bit integer giving 256 possible different shades of grey from black to white. If the levels are evenly spaced then the difference between successive greylevels is significantly better than the greylevel resolving power of the human eye.

Greyscale images are very common, in part because much of today's image capture hardware can only support 8-bit images. In addition, greyscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process colour images.

High-Pass Filter

A spatial filter that emphasises the areas of higher frequency (transitions from

dark to light and light to dark) or attenuates those areas with low frequency. It emphasises the high frequency details in an image. (cf. Section 2.4.4)

Histogram

A statistical analysis of the number of pixels at each brightness level in an image.

Histogram Equalisation

Re-mapping the intensity values in an image to new values so that a histogram of the new image contains approximately the same number of pixels at each intensity value.

Hue

The attribute of a colour explaining what colour it is. (purple, green, red, etc.)

Idempotence

Some operators have the special property that applying them more than once to the same image produces no further change after the first application. Such operators are said to be idempotent. Examples include the morphological operators opening and closing.

IIR (Infinite Impulse Response) Filter

A temporal filtering technique that feeds back a percentage of the previous result to be averaged in with the current frame. This has the effect of attenuating vibrations, motion, and spurious noise in the frame.

Intensity

The brightness value assigned to a pixel in a digital image.

Isotropic Operator

An isotropic operator in an image processing context is one which applies equally well in all directions in an image, with no particular sensitivity or bias towards one particular set of directions (e. g. compass directions). A typical example is the zero crossing edge detector which responds equally well to edges in any orientation. Another example is Gaussian smoothing. It should be borne in mind that although an operator might be isotropic in theory, the actual implementation of it for use on a discrete pixel grid may not be perfectly isotropic. An example

of this is a Gaussian smoothing filter with very small standard deviation on a square grid.

Kernel

A kernel is a (usually) smallish matrix of numbers that is used in image convolutions. Differently sized kernels containing different patterns of numbers give rise to different results under convolution. For instance, Fig. 126 shows a 3×3 kernel that implements a mean filter. The word 'kernel' is also commonly used

1	1	1
1	1	1
1	1	1

Set of coordinate points :
{ (-1,-1), (0,-1), (1,-1),
(-1,0), (0,0), (1,0),
(-1,1), (0,1), (1,1) }

Figure 126: Convolution kernel for a mean filter with 3×3 neighbourhood.

as a synonym for 'structuring element', which is a similar object used in mathematical morphology. A structuring element differs from a kernel in that it also has a specified origin.

Laplacian Filter

An omni-directional spatial edge enhancement filter.

Light

Electromagnetic radiation in the spectral range detectable by the human eye (approx. 380 to 720 nm).

Logical Operator

Logical operators are generally derived from Boolean algebra, which is a mathematical way of manipulating the truth values of concepts in an abstract way without bothering about what the concepts actually mean. The truth value of a concept in Boolean value can have just one of two possible values: true or false. Boolean algebra allows to represent things like:

The block is both red and large

by something like:

A AND B

where A represents 'The block is red', and B represents 'The block is large'. Now each of these sub-phrases has its own truth value in any given situation: each sub-phrase is either true or false. Moreover, the entire composite phrase also has a truth value: it is true if both of the sub-phrases are true, and false in any other case. This AND combination rule (and its dual operation NAND) can be written using a truth-table as shown in Fig. 127, in which conventionally true is represented by 1, and false by zero. The left hand table shows each of the

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	1

NAND

Figure 127: Truth-tables for AND and NAND.

possible combinations of truth values of A and B, and the resulting truth value of A AND B. Similar truth-tables can be set up for the other logical operators: NAND, OR, NOR, XOR, XNOR and NOT.

Turning now to an image processing context, the pixel values in a binary image, which are either 0 or 1, can be interpreted as truth values as above. Using this convention we can carry out logical operations on images simply by applying the truth-table combination rules to the pixel values from a pair of input images (or a single input image in the case of NOT). Normally, corresponding pixels from each of two identically sized binary input images are compared to produce the output image, which is another binary image of the same size. As with other image arithmetic operations, it is also possible to logically combine a single input image with a constant logical value, in which case each pixel in the input image is compared to the same constant in order to produce the corresponding output pixel.

Logical operations can also be carried out on images with integer pixel values. In this extension the logical operations are normally carried out in bitwise fashion

on binary representations of those integers, comparing corresponding bits with corresponding bits to produce the output pixel value.

Look-up Table and Colourmap

Look-Up Tables or LUTs are fundamental to many aspects of image processing. An LUT is simply a table of cross-references linking index numbers to output values. The most common use is to determine the colours and intensity values with which a particular image will be displayed, and in this context the LUT is often called simply a colourmap. See also 'Colour Images'.

As well as their use in colourmaps, LUTs are often used to remap the pixel values within an image. This is the basis of many common image processing point operations such as thresholding, gamma correction and contrast stretching. The process is often referred to as anamorphosis.

Low-Pass Filter

A filter that attenuates high frequency detail in an image. (cf. Section 2.4.3)

Masking

A mask is a binary image consisting of zero- and non-zero values. If a mask is applied to another binary or to a greyscale image of the same size, all pixels which are zero in the mask are set to zero in the output image. All others remain unchanged.

Masking can be implemented either using pixel multiplication or logical AND, the latter in general being faster.

Masking is often used to restrict a point or arithmetic operator to an area defined by the mask. We can, for example, accomplish this by first masking the desired area in the input image and processing it with the operator, then masking the original input image with the inverted mask to obtain the unprocessed area of the image and finally recombining the two partial images using image addition. See also 'Region of Interest'.

Mathematical Morphology

The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set

theory. The operators are particularly useful for the analysis of binary images and common usages include edge detection, noise removal, image enhancement and image segmentation.

The two most basic operations in mathematical morphology are erosion and dilation. Both of these operators take two pieces of data as input: an image to be eroded or dilated, and a structuring element (also known as a kernel). The two pieces of input data are each treated as representing sets of coordinates in a way that is slightly different for binary and greyscale images.

For a binary image, white pixels are normally taken to represent foreground regions, while black pixels denote background. (Note that in some implementations this convention is reversed, and so it is very important to set up input images with the correct polarity for the implementation being used). Then the set of coordinates corresponding to that image is simply the set of two-dimensional Euclidean coordinates of all the foreground pixels in the image, with an origin normally taken in one of the corners so that all coordinates have positive elements.

For a greyscale image, the intensity value is taken to represent height above a base plane, so that the greyscale image represents a surface in three-dimensional Euclidean space. Fig. 128 shows such a surface. Then the set of coordinates associated with this image surface is simply the set of three-dimensional Euclidean coordinates of all the points within this surface and also all points below the surface, down to the base plane.[2] The structuring element is already just a set of point coordinates (although it is often represented as a binary image). It differs from the input image coordinate set in that it is normally much smaller, and its coordinate origin is often not in a corner, so that some coordinate elements will have negative values.[3]

Binary morphology can be seen as a special case of greylevel morphology in which the input image has only two greylevels at values 0 and 1.

[2]Note that even when only points with integer coordinates are being considered, this is a lot of points, so usually algorithms are employed that do not need to consider all the points.

[3]Note that in many implementations of morphological operators, the structuring element is assumed to be a particular shape (e. g. a 3×3 square) and so is hardwired into the algorithm.

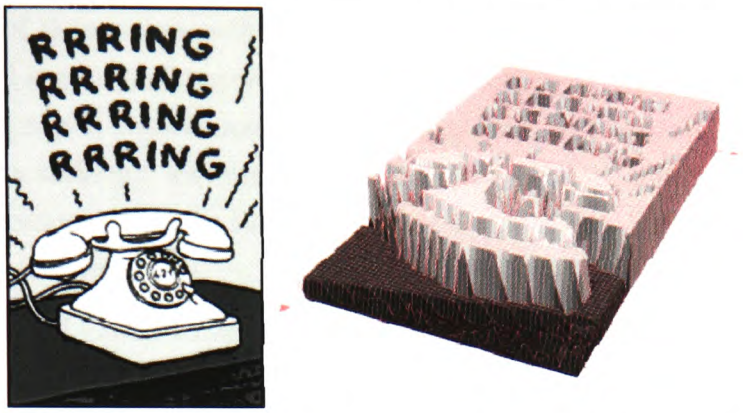


Figure 128: Simple greylevel image and the corresponding surface in three-dimensional Euclidean space.

Erosion and dilation work (at least conceptually) by translating the structuring element to various points in the input image, and examining the intersection between the translated kernel coordinates and the input image coordinates. For instance, in the case of erosion, the output coordinate set consists of just those points to which the origin of the structuring element can be translated, while the element still remains entirely 'within' the input image.

Virtually all other mathematical morphology operators can be defined in terms of combinations of erosion and dilation along with set operators such as intersection and union. Some of the more important are opening, closing and skeletonisation.

Median Filter

A filter that calculates the median value of pixels in a neighbourhood and then assigns that value to the result pixel.

Morphological Mask

Also called a structuring element. A set of logical values (for a binary image) or grey-level values used in binary or grey-scale morphology.

Morphology

See 'Mathematical Morphology'.

Multi-spectral Image

A multi-spectral image is a collection of several monochrome images of the same

scene, each of them taken with a different sensor. Each image is referred to as a band. A well known multi-spectral (or multi-band image) is a RGB colour image, consisting of a red, a green and a blue image, each of them taken with a sensor sensitive to a different wavelength. In image processing, multi-spectral images are most commonly used for Remote Sensing applications. Satellites usually take several images from frequency bands in the visual and non-visual range.[4]

All the standard single-band image processing operators can also be applied to multi-spectral images by processing each band separately. For example, a multi-spectral image can be edge detected by finding the edges in each band and than ORing the three edge images together. However, more reliable edges will be obtained, if a pixel is associated with an edge based on its properties in all three bands and not only in one.

To fully exploit the additional information which is contained in the multiple bands, the images should be considered as one multi-spectral image rather than as a set of monochrome greylevel images. For an image with k bands, the brightness of each pixel can then be described as a point in a k-dimensional space represented by a vector of length k.

Special techniques exist to process multi-spectral images. For example, to classify a pixel as belonging to one particular region, its intensities in the different bands are said to form a feature vector describing its location in the k-dimensional feature space. The simplest way to define a class is to choose a upper and lower threshold for each band, thus producing a k-dimensional 'hyper-cube' in the feature space. Only if the feature vector of a pixel points to a location within this cube, is the pixel classified as belonging to this class.

The disadvantage of multi-spectral images is that the required computation time and memory increase significantly. However, since the speed of the hardware will increase and the costs for memory will decrease in the future, it can be expected that multi-spectral images will become more important in many fields of computer vision.

[4]LANDSAT 5, for example, produces 7 band images with the wavelength of the bands being between 450 and 1250 nm.

Multitone

Colourised greyscale image. (mathematical)

Neighbourhood Squaring Element

Element that calculates the sum of squares of all the pixels in a given neighbourhood.

Neighbourhood

A pixel and those surrounding pixels whose values are to be used in a processing operation that calculates a new value for the pixel.

Non-Linear Filter

Suppose that an image processing operator F acting on two input images A and B produces output images C and D respectively. If the operator F is linear, then

$$F(a \times A + b \times B) = a \times C + b \times D \tag{73}$$

where a and b are constants. In practice, this means that each pixel in the output of a linear operator is the weighted sum of a set of pixels in the input image.

By contrast, non-linear operators are all the other operators. For example, the threshold operator is non-linear, because individually, corresponding pixels in the two images A and B may be below the threshold, whereas the pixel obtained by adding A and B may be above threshold. Similarly, an absolute value operation is non-linear:

$$|-1+1| \neq |-1| + |1| \tag{74}$$

as is the exponential operator:

$$\exp(1+1) \neq \exp(1) + \exp(1) \tag{75}$$

Nyquist Sampling Theorem

A theory of sampling that says to accurately represent the details in a continuous (analog) image signal, the sampling rate must be at least twice as fast as the highest spatial frequency in the image.

Opening

Binary or grey-scale morphological operation that is accomplished by performing an erosion operation followed by a dilation operation. This cleans up the

image by removing small bright spikes or noise and then returning the remaining objects to their original size.

Pattern Matching

Comparing an image with a pattern template and then returning the location(s) where the pattern is found. See also 'Correlation'.

Pixel

In order for any digital computer processing to be carried out on an image, it must first be stored within the computer in a suitable form that can be manipulated by a computer program. The most practical way of doing this is to divide the image up into a collection of discrete (and usually small) cells, which are known as pixels[5]. Most commonly, the image is divided up into a rectangular grid of pixels, so that each pixel is itself a small rectangle. Once this has been done, each pixel is given a pixel value that represents the colour of that pixel. It is assumed that the whole pixel is the same colour, and so any colour variation that did exist within the area of the pixel before the image was discretised is lost. However, if the area of each pixel is very small, then the discrete nature of the image is often not visible to the human eye.[6]

Other pixel shapes and formations can be used, most notably the hexagonal grid, in which each pixel is a small hexagon. This has some advantages in image processing, including the fact that pixel connectivity is less ambiguously defined than with a square grid, but hexagonal grids are not widely used. Part of the reason is that many image capture systems (e. g. most CCD cameras and scanners) intrinsically discretise the captured image into a rectangular grid in the first instance.

Pixel Connectivity

The notation of pixel connectivity describes a relation between two or more pixels. For two pixels to be connected they have to fulfil certain conditions on the pixel brightness and spatial adjacency.

First, in order for two pixels to be considered connected, their pixel values must both be from the same set of values V. For a greyscale image, V might be any

[5]Pixel is a short form for Picture Element

[6]In actual fact the human eye discretises the captured images, too. See Section 2.2.

range of greylevels, e. g. $V = \{24, 25, \ldots, 42\}$, for a binary image it simply is $V = \{1\}$.

To formulate the adjacency criterion for connectivity, at first the notation of neighbourhood is introduced. For a pixel p with the coordinates (x, y) the set of pixels given by:

$$N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\} \tag{76}$$

is called its 4-neighbours. Its 8-neighbours are defined as

$$N_8(p) = N_4(p) \cup \{(x+1, y+1), (x-1, y+1), (x-1, y+1), (x-1, y-1)\} \tag{77}$$

From this the definition for 4- and 8-connectivity can be inferred:

Two pixels p and q, both having values from a set V are 4-connected if q is from the set $N_4(p)$ and 8-connected if q is from $N_8(p)$.

General connectivity can either be based on 4- or 8-connectivity; for the following discussion 4-connectivity is used.

A pixel p is connected to a pixel q if p is 4-connected to q or if p is 4-connected to a third pixel which itself is connected to q. Or, in other words, two pixels q and p are connected if there is a path from p and q on which each pixel is 4-connected to the next one.

A set of pixels in an image which are all connected to each other is called a connected component. Finding all connected components in an image and marking each of them with a distinctive label is called connected component labelling, or blob labelling for binary images.

An example of a binary image with two connected components which are based on 4-connectivity can be seen in Fig. 129. If the connectivity were based on 8-neighbours, the two connected components would merge into one. See also 'Blob Analysis'.

Pixel Values

Each of the pixels that represents an image stored inside a computer has a pixel value which describes how bright that pixel is, and/or what colour it should be. In the simplest case of binary images, the pixel value is a 1-bit number

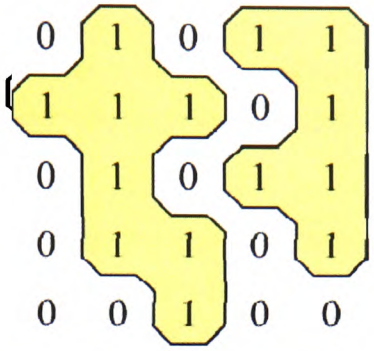


Figure 129: Two connected components based on 4-connectivity.

indicating either foreground or background. For a greyscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of grey.

To represent colour images, separate red, green and blue components must be specified for each pixel (assuming an RGB colourspace), and so the pixel 'value' is actually a vector of three numbers. Often the three different components are stored as three separate 'greyscale' images known as colour planes (one for each of red, green and blue), which have to be recombined when displaying or processing.

Multi-spectral images can contain even more than three components for each pixel, and by extension these are stored in the same kind of way, as a vector pixel value, or as separate colour planes.

The actual greyscale or colour component intensities for each pixel may not actually be stored explicitly. Often, all that is stored for each pixel is an index into a colourmap in which the actual intensity or colours can be looked up.

Although simple 8-bit integers or vectors of 8-bit integers are the most common sorts of pixel values used, some image formats support different types of value, for instance 32-bit signed integers or floating point values. Such values are extremely useful in image processing as they allow processing to be carried out on the image

where the resulting pixel values are not necessarily 8-bit integers. If this approach is used then it is usually necessary to set up a colourmap which relates particular ranges of pixel values to particular displayed colours.

Primary Colour

It is a useful fact that the huge variety of colours that can be perceived by humans can all be produced simply by adding together appropriate amounts of red, blue and green colours. These colours are known as the primary colours. Thus in most image processing applications, colours are represented by specifying separate intensity values for red, green and blue components. This representation is commonly referred to as RGB.

The primary colour phenomenon results from the fact that humans have three different sorts of colour receptors in their retinas which are each most sensitive to different visible light wavelengths.

The primary colours used in painting (red, yellow and blue) are different. When paints are mixed, the 'addition' of a new colour paint actually subtracts wavelengths from the reflected visible light.

Rank Value Filter

A neighbourhood process where each source neighbourhood of pixels is ranked in order of intensity, and the result pixel is the mean, median, minimum or maximum pixel in the neighbourhood. This method is often used to remove noise.

Resolution

The number of bits used to represent the intensity value of a pixel.

Region of Interest

User or process defined area, usually rectangular, which is used for further processing.

RGB and Colourspace

A colour perceived by the human eye can be defined by a linear combination of the three primary colours red, green and blue. These three colours form the basis for the RGB-colourspace. Hence, each perceivable colour can be defined

by a vector in the three-dimensional colourspace. The intensity is given by the length of the vector, and the actual colour by the two angles describing the orientation of the vector in the colourspace.

Sampling

Taking sample values from a continuous analogue stream at discrete intervals.

Sampling Theorem

See 'Nyquist Sampling Theorem'.

Saturation

The attribute of a colour explaining how far away from grey of the same lightness the colour is.

Segmentation

Dividing an image into discrete objects and background.

Sensor

A device that translates light or heat or other physical phenomena into analogue or digital data. In image processing usually a CCD or CMOS camera chip is the sensor.

Sobel Filter

An omni-directional edge detection filter which is accomplished by first adding together the results of a horizontal filter and a vertical and then taking the square roots of those values.

Spatial

Referring to the two dimensional nature of an image.

Spatial Aliasing

The undesirable effects that occur when an image is sampled at less than twice the rate of the highest spatial frequency of the image.

Spatial Domain

For simplicity, assume that the image I being considered is formed by projection from scene S (which might be a two- or three-dimensional scene, etc.).

The spatial domain is the normal image space, in which a change in position in I directly projects to a change in position in S. Distances in I (in pixels) correspond to real distances (e. g. in meters) in S.

This concept is used most often when discussing the frequency with which image values change, that is, over how many pixels does a cycle of periodically repeating intensity variations occur. One would refer to the number of pixels over which a pattern repeats (its periodicity) in the spatial domain.

In most cases, the Fourier Transform will be used to convert images from the spatial domain into the frequency domain.

Spatial Derivative

The spatial derivative refers to how much the image intensity values change per change in image position.

Spatial Filter

A filter that operates in the spatial domain as opposed to the frequency domain to accentuate or attenuates the appearance of the spatial details, for example the transitions of intensity in an image.

Spatial Frequency

The spatial frequency refers to the periodicity with which the image intensity values change. Image features with high spatial frequency (such as edges) are those that change greatly in intensity over short image distances.

Spatial Resolution

The number of pixels in the horizontal and vertical dimensions used to represent a digital image.

Structuring Element

The field of mathematical morphology provides a number of important image processing operations, including erosion, dilation, opening and closing. All these morphological operators take two pieces of data as input. One is the input image, which may be either binary or greyscale for most of the operators. The other is the structuring element. It is this that determines the precise details of the effect of the operator on the image.

The structuring element[7] consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. Normally cartesian coordinates are used and so a convenient way of representing the element is as a small image on a rectangular grid. Fig. 130 shows a number of different structuring elements of various sizes. In each case the origin is marked by a ring around that point. The origin does not have to be in the centre of the structuring element, but often it is. As suggested by the Fig. 130, structuring elements that fit into a 3×3 grid with its origin at the centre are the most commonly seen type. Note that each

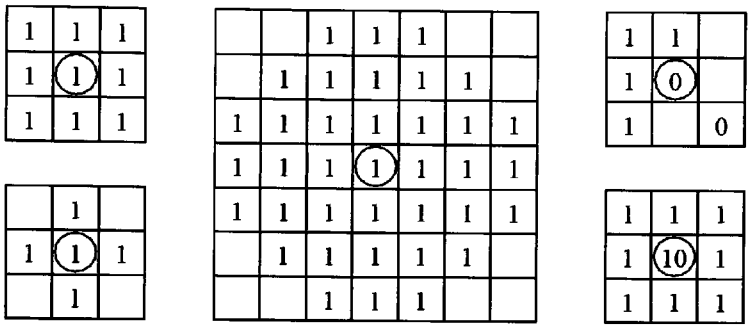


Figure 130: Some example structuring elements.

point in the structuring element may have a value. In the simplest structuring elements used with binary images for operations such as erosion, the elements only have one value, conveniently represented as a one. More complicated elements, such as those used with thinning or greyscale morphological operations, may have other pixel values.

An important point to note is that although a rectangular grid is used to represent the structuring element, not every point in that grid is part of the structuring element in general. Hence the elements shown in Fig. 130 contain some blanks.

When a morphological operation is carried out, the origin of the structuring element is typically translated to each pixel position in the image in turn, and then the points within the translated structuring element are compared with the underlying image pixel values. The details of this comparison, and the effect of the outcome depend on which morphological operator is being used.

[7]The structuring element is sometimes called the kernel.

Subtractive Primaries

Cyan, Magenta and Yellow. When all 3 colours are combined at 100% on white paper, black is produced. When combining different variations of these colours, a wide gamut of colours will be obtained.

Temporal Aliasing

The undesirable effect that occurs in motion sequences when the frame rate is not high enough for the sampling rate. For example, the spokes of a wheel may appear to turn backwards if the frame rate of the video input is not fast enough to capture the motion of the wheel.

Thresholding

The assigning of a binary value to a pixel based on whether its intensity falls below, or above a threshold value.

Unsharp Masking

A filter used to reduce noise while retaining most of the high frequency detail in an image. Unsharp masking is accomplished by subtracting a low-pass filtered version of an image from the original image.

Acronyms, Initialisms, Abbreviations and Portmanteaus

Explanation of the acronyms[8], initialisms[9], abbreviations[10] and portmanteaus[11] used in the book or common to the subject.

A/D	Analogue to Digital
ADC	Analogue to Digital Converter
ADALINE	ADAptive LINear Elements
AFC	adaptive fuzzy-c-varieties
AI	Artifical Intelligence
ANN	Artificial Neural Network
ANSI	American National Standards Institute
AO	Automatic Optimisation
AOS	Automatic-Optimisation-System
ART	Adaptive Resonance Theory
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
AV	Artificial Vision
BAUD	Bits At Unit Density
BASIC	Beginner's All-Purpose Symbolic Instruction Code
BDR	Border Detection Routine
BIOS	Basic Input/Output System

[8] An acronym is formed by the initial letter or letters of the major parts of a phrase. The resulting group of letters must be pronounceable. (E. g. BIOS, ANSI)

[9] An initialism is formed like an acronym, but not pronounceable. (E. g. CCD, STFT)

[10] An abbreviation is a shorted form of a word. (E. g. Fig., Chap.)

[11] A portmanteau is a word formed by combining, or blending, two other words. (E. g. Modem, Pixel)

Bit	Basic Information Unit *or* Binary Digit (also kBit and MBit for 1024 and 1.048.576 Bit)
BLOB	Binary Large OBject
BMP	Basic Multilingual Plane
BNN	Biological Neural Network
B/W	black and white
Byte	Binary Yoked Transfer Element (also kByte and MByte for 1024 and 1.048.576 Byte)
CBF	Cubic Basis Function
CC	Cascade Correlation
CCD	Charged-Coupled Device
Chap.	Chapter, plural Chaps.
CIE	Commission internationale de l'Eclairage (International Commission on Illumination)
CIE Lab	CIE Lab colour model
CIE L*a*b*	CIE L*a*b* colour model
CIE Luv	CIE Luv colour model
CIE XYZ	CIE XYZ colour model
CISC	Complex Instruction Set Computer
CM	Cooccurrence Matrix
CMOS	Complementary Metal-Oxide Semiconductor
CMYK	Cyan, Magenta, Yellow and Black (ink colours for printing)
COA	Centre of Area
CPU	Central Processing Unit
CS	Core-System
CT	Chaos Theory
CWT	Continuous Wavelet Transformation
D/A	Digital to Analog
DAC	Digital to Analogue Converter
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNA	Desoxyribonucleic Acid
dpi	dots per inch
DSP	Digital Signal Processor
DST	Discrete Sine Transform

DU	Data Unit
DWT	Discrete Wavelet Transformation
EA	Evolutionary Algorithm
EC	Evolutionary Computing
EP	Evolutionary Programming
Eq.	Equation, plural Eqs.
ES	Enhanced-System
FCA	Fuzzy Cluster Analysis
FC	Fuzzy Clustering
FCM	fuzzy-c-means
FCQS	fuzzy-c-quadratic-shells
FCR	fuzzy-c-rings
FCRS	fuzzy-c-rectangular-shells
FCS	fuzzy-c-shells
FCS	Fuzzy Clustering System
FCSS	fuzzy-c-spherical-shells
FCV	fuzzy-c-varieties
FFT	Fast Fourier Transformation
FHT	Fast Hadamard Transform
Fig.	Figure, plural Figs.
FIR	Finite Impulse Response (Filter)
FL	Fuzzy Logic
FLOPS	Floating Point Operations Per Second
FNN	Fuzzy Neural Networks
FPU	Floating Point Unit
FPU	Fuzzy Processing Unit
FT	Fourier Transformation
GA	Genetic Algorithm
GG	Gath-Geva
GIF	Graphic Interchange Format
GK	Gustafson-Kessel
GMD	Gaussian mixture decomposition
HCL	Hue, Chroma and Luminance (colour model)
HIS	Hue, Intensity and Saturation (colour model)
HSB	Hue, Saturation and Brightness (colour model)

HSL	Hue, Saturation and Luminance (colour model)
HSV	Hue, Saturation and Value (colour model)
HTML	Hyper-Text Markup Language
Hz	Hertz (frequency unit equal to one cycle per second, also kHz and MHz for Kilo- and Megahertz)
IC	Integrated Circuit
IEE	The Institution of Electrical Engineers
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
IHS	intensity, hue (average wavelength) and saturation
IIR	Infinite Impulse Response (Filter)
I/O	Input/Output
IR	infra-red
ISO	International Organization for Standardization
JavaNNS	Java Neuronale Netze Simulator
JPEG	Joint Photographic Experts Group
LAT	Local Adaptive Threshold
LCM	Logarithmic Cooccurrence Matrix
LoG	Laplacian of Gaussian
LUT	Look-Up Table
LVQ	Learning Vector Quantisation
MADALINE	Multiple ADAptive LINear Elements
MCU	Micro Controller Unit
MDCM	Multi-Dimensional Cooccurrence Matrix
MIQ	Machine Intelligence Quotient
MSA	Multi-Scale Algorithm
NF	Neuro Fuzzy
NIR	near-infra-red
NN	Neural Network
NTSC	National Television Systems Committee (US Colour TV standard)
OCR	Optical Character Recognition
PAL	Phase Alternating Line system (European colour TV standard)
PC	Personal Computer
PCA	Pyramidal Coding Algorithm
PI	Proportional-Integral
PID	Proportional-Integral-Differential

Pixel	Picture Element, one image point (also kPixel and MPixel for 1024 and 1.048.576 Pixel)
PNG	Portable Network Graphics
PNN	Probabilistic Neural Network
QC	Quality Control
RBF	Radial Basis Function
RGB	CIE spectral primary system
RISC	Reduced Instruction Set Computer
rms	root mean square
ROI	Region of Interest
SC	Soft-Computing
Sec.	Section, plural Secs.
SECAM	Sequential Couleur A'Memorie (French and Russian Colour TV Standard)
SIMD	Single Instructions Multiple Data
SNNS	Stuttgarter Neuronale Netze Simulator
SOM	Self-Organised Maps
STFT	Short-Time Fourier Transformation
Tbl.	Table, plural Tbls.
TCS	Texture Classification System
TEM	Texture Energy Measures
TIFF	Tagged Image File Format
USM	UnSharp Masking
UV	Ultra Violet
VDI	Verein Deutscher Ingenieure
VLSI	Very Large Scale Integration
WCM	Wide Cooccurrence Matrix
WT	Wavelet Transformation
X GUI	X Graphical User Interface
YIQ	colour components in NTSC colour space
Yuv	CIE uniform chromaticity scale (colour components in SECAM and PAL colour spaces)

References

[Abe91] Dirk Abel. Fuzzy Control – eine Einführung ins Unscharfe. *at – Automatisierungstechnik*, 39(12):433–438, Dezember 1991. ISSN 0178-2312.

[Abm94] Wolfgang Abmayr. *Einführung in die digitale Bildverarbeitung.* B. G. Teubner, Stuttgart, 1994.

[Aok90] S. Aoki. Application of fuzzy control logic for dead-time processes in a glass melting furnance. *Fuzzy Sets and Systems*, 38(2):251–265, 1990. ISSN 0165-0114.

[AW91] R. J. Ahlers and H. J. Warnecke. *Industrielle Bildverarbeitung.* Addison-Wesley, Bonn, München, 1991.

[Bäh85] H.-P. Bähr. *Digitale Bildverarbeitung: Anwendung in Photogrammetrie und Fernerkundung.* Herbert Wichmann Verlag, Karlsruhe, 1985.

[Bar04] Xavier Lladó Bardera. *Texture recognition under varying imaging geometries.* PhD thesis, Universitat de Girona, Girona, Spain, 2004.

[Bay89] Thomas Bayer. Segmentierung von verklebten Einzelzeichen. Technical report, AEG Forschungsinstitut Ulm, 1989. Technischer Bericht Nr. 12.005/89.

[BB82] Dana Harry Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, 1982.

[BB93] Henning Bässmann and Philipp W. Besslich. *Bildverarbeitung Ad Oculos.* Springer-Verlag, Berlin, 1993.

[BB97] Matthias Blume and Dan R. Ballard. Image annotation based on Learning Vector Quantization and localised Haar wavelet transform features. *SPIE Applications and Science of Neural Networks Conference*, 1997. Paper 3077-22.

[BBE95] Matthias Blume, Daniel A. Van Blerkom, and Sadik C. Esener. Fuzzy ARTMAP modifications for intersecting class distributions. Technical report, Department of Electrical and Computer Engineering, University of California at San Diego, 1995.

[BE95] Matthias Blume and Sadik C. Esener. Optoelectronic fuzzy ARTMAP processor. *Optical Computing*, 10:213–215, March 1995.

[BE96] Matthias Blume and Sadik C. Esener. An efficient mapping of fuzzy ART onto a neural architecture. Technical report, Department of Electrical and Computer Engineering, University of California at San Diego, 1996.

[Bez73] James C. Bezdek. *Fuzzy Mathematics in Pattern Classification.* PhD thesis, Cornell University, 1973.

[Bez81] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York/London, 1981. ISBN 0-306-40671-3.

[Bez93] James C. Bezdek. Fuzzy models and digital signal processing (for pattern recognition): Is this a good marriage? *Digital Signal Processing*, 3:253–270, 1993.

[BG92] Hans Bandemer and Siegfried Gottwald. *Einführung in Fuzzy-Methoden.* Akademie Verlag, Berlin, 1992. ISBN 3-05-501458-8.

[Bla95] Benjamin Blankertz. Einführung in Neuronale Netze. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1995.

[BO02] M.K. Bashar and N. Ohnishi. A block processing approach (bpa) to texture classification using cortex transform. In *Pattern Recognition in Remote Sensing (PRRS 2002)*, Niagara Falls, Canada, 2002. BMVA Press. ISBN 1-901725-18-9.

[Böh93] Gert Böhme. *Fuzzy-Logik.* Springer-Verlag, Berlin/Heidelberg, 1993.

[Bös95] Klaus D. Bösing. Über Aspekte der visuellen Wahrnehmung und der optischen Eigenschaften idealer Oberflächen. Technical Report 87/12, Institut für Technische Informatik, Technische Universität Berlin, 1995.

[Bot93] Hans-Heinrich Bothe. *Fuzzy Logic – Einführung in Theorie und Anwendungen.* Springer-Verlag, Berlin, 1993. ISBN 3-540-56166-8.

[Bro66] Phil Brodatz. *Textures – A Photographic Album for Artists & Designers.* Dover Publications, Mineola, New York, 1966.

[Bro71] Phil Brodatz. *Wood and Wood Grains – A Photographic Album for Artists & Designers.* Dover Publications, New York, 1971.

[Bro76] Phil Brodatz. *Land, Sea & Sky – A Photographic Album for Artists & Designers.* Dover Publications, New York, 1976.

[Bro95] Dirk Brosowski. Fuzzy Kohonen Netze. Lehrstuhl für Informatik, WWU Münster, November 1995.

[BS95] C. Busch and M. Schmerer. Ein Verfahren zur Texturanalyse basierend auf multiplen Waveletbasen. In G. Sagerer, editor, *Mustererkennung 1995*, GI Informatik aktuell, pages 562–569. Springer Verlag, Berlin/Heidelberg/New York, 1995.

[BW94] D. Blacknell and R. G. White. Optimum classification of non-Gaussian process using neural networks. *IEE Proc.-Vis. Image Signal Process.*, 141(1):56–66, January 1994.

[Cal91] David L. Calloway. Constructing an optimal binary phase-only filter using a genetic algorithm. *SPIE Optical Information Processing Systems and Architecture III*, 1564:395–402, 1991.

[CANS03] Rand C. Chandler, A. Antonio Arroyo, M. Nechyba, and E. M. Schwartz. Robot navigation and textural analysis. In *Florida Conference on Recent Advances in Robotics (FCRAR 2003)*, Dania Beach, Florida, 5 2003.

[cdp] cdphysik. ULR of the Physics and Astronomy Department of the University Heidelberg, Germany. `http://www.physik.uni-heidelberg.de/cdphysik`.

[CFW94] V. Cherkassky, J. H. Friedmann, and H. Wechsler. *From Statistics to Neural Network.* Springer Verlag, 1994.

[CG87a] Gail A. Carpenter and Stephen Grossberg. ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 1987.

[CG87b] Gail A. Carpenter and Stephen Grossberg. ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3, 1987.

[CG87c] Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, (37), 1987.

[CG88] Gail A. Carpenter and Stephen Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, March 1988.

[CGM+92] Gail A. Carpenter, Stephen Grossberg, Natalya Markuzon, John Reynolds, and David B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(4):698–713, September 1992.

[CGR91a] Gail A. Carpenter, Stephen Grossberg, and John H. Reynolds. A self-organizing ARTMAP neural architecture for supervized learning and pattern recognition. In R. J. Mammone and Y. Zeevi, editors, *Neural Networks*, pages 43–80. Academic Press, 1991.

[CGR91b] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. ART-2A: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 4:493–504, 1991.

[CGR91c] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns in an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[CH80] Richard W. Conners and Charles A. Harlow. A theoretical comparison of texture algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(3):204–222, May 1980.

[CHdJ92] G. S. Cox, F. J. Hoare, and G. de Jager. Experiments in lung cancer nodule detection using texture analysis and neural network classifiers. In *3rd Soth African Workshop on Pattern Recognition*, Pretoria, 1992.

[CHL98] Lili Chen, Jun Hong, and Baohong Li. Classification of multipolarized SAR images by an unsupervised back-propagation neural network with texture discrimination. In Ji Zhou and Anil K. Jain and, editors, *International Symposium on Multispectral Image Processing (ISMIP'98)*, volume 3545, pages 478–481, Bellingham, Washington, 1998. SPIE–The International Society for Optical Engineering. ISBN 0-8194-3006-4.

[Chr92] Gerhard Chroust. *Modelle der Software-Entwicklung – Aufbau und Interpretation von Vorgehensmodellen.* Oldenbourg Verlag, München/Wien, 1992.

[CJ96a] David A. Clausi and M. Ed Jernigan. A fast method to determine coocurrence texture features using a linked list implementation. Technical report, Department of Geography (Earth Observation Laboratory), University of Waterloo, Canada, 1996?

[CJ96b] David A. Clausi and M. Ed Jernigan. Towards a novel approach for texture segmantation of SAR sea ice imagery. Technical report, Department of Geography (Earth Observation Laboratory), University of Waterloo, Canada, 1996?

[CLZ81] Gerhard Czihak, Helmut Langer, and Hubert Ziegler. *Biologie.* Springer-Verlag, Berlin, 1981. ISBN 3-540-56003-3.

[CM87] Fergus W. Campbell and Lamberto Maffei. Kontrast und Raumfrequenz. In Manfred Ritter, editor, *Wahrnehmung und visuelles Sehen.* Spektrum der Wissenschaft, 1987.

[CMD+95] A. Collignon, F. Maes, D. Delaere, D. Vandermeulen, P. Suetens, and G. Marchal. Automated multi-modality image registration based on information theory. In *Information Processing in Medical Imaging*, pages 262–274, 1995.

[Cox92] Earl Cox. Integrating fuzzy logic into neural nets. *AI Expert*, pages 42–47, June 1992.

[Dav89] R. N. Dave. Use of the adaptive fuzzy clustering algorithm to detect lines in digital images. *Intelligent Robots and Computer Vision*, VIII(1192 (2)):600–611, 1989.

[Dav91] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[DG88] Ingrid Daubechies and Alex Grossmann. Frames in the bargmann space of entire-functions. In *Communications on Pure and Applied Mathematics*, volume 41(2), pages 151–164, 1988.

[DG02] Fabio DelliAcqua and Paolo Gamba. Spatial analysis of satellite SAR for urban applications. In *Pattern Recognition in Remote Sensing (PRRS 2002)*, Niagara Falls, Canada, 2002. BMVA Press. ISBN 1-901725-18-9.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[DMW92] Ingrid Daubechies, Stéphane Mallat, and Alan S. Willsky. Special issue on wavelet transforms and multiresolution signal analysis – introduction. In *IEEE Transactions on Information Theory*, volume 38, pages 529–531, 1992.

[Don92] David L. Donoho. Wavelet shrinkage and W.V.D.: A 10-minute tour. Technical report, Stanford University, June 1992.

[Don95] David L. Donoho. De-noising by soft-thresholding. In *IEEE Transactions on Information Theory*, volume 41(3), pages 613–627, 1995.

[DP78] Didier Dubois and Henri Prade. Operations on fuzzy numbers. *International Journal of Systems Science*, 9(6):613–626, June 1978. ISSN 0020-7721.

[DP80] Didier Dubois and Henri Prade. *Fuzzy Sets and Systems – Theory and Applications*. Academic Press, New York, 1980. ISBN 0-12-222750-6.

[Dra94] John Drakopoulos. Probabilities, possibilities, and fuzzy sets. Technical report, Department of Computer Science, Knowledge Systems Laboratory, Stanford University, January 1994.

[Dun73] Joseph C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[Dus99] Thorsten Duscha. Morphological image processing and neural networks. Technical report, Fachhochschule Braunschweig/Wolfenbüttel, 1999.

[dVA99] Juan Martínez-Cabeza de Vaca-Alajarín. Marble slabs quality classification system using texture recognition and neural networks methodology. In *European Symposium on Artificial Neural Networks (ESANN'1999)*, pages 75–80, Bruges, Belgium, 1999. ISBN 2-600049-9-X.

[EN95] A. N. Evans and M. S. Nixon. Mode filtering to reduce ultrasound speckle for feature extraction. *IEE Proc.-Vis. Image Signal Process.*, 142(2):87–94, April 1995.

[EP93] I. M. Elfadel and R. W. Picard. Gibbs random fields, co-occurrences, and texture modeling. Technical report, MIT Media Laboratory Perceptual Computing Group, January 1993. Technical Report #204.

[Ern91] Hartmut Ernst. *Einführung in die digitale Bildverarbeitung*. Franzis-Verlag, München, 1991.

[FA91] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[Fab94] Vance Faber. Clustering and the continuous k-means algorithm. Computer research group technical report, Los Alamos National Laboratory, 1994.

[Fah88a] Scott E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical report, School of Computer Science, Carnegie Mellon University, September 1988. 88-162.

[Fah88b] Scott E. Fahlman. The recurrent cascade-correlation architecture. Technical report, School of Computer Science, Carnegie Mellon University, May 1988. 91-100.

[Fel04] Niko Felekidis. Erkennung und Verfolgung von Kraftfahrzeugen in Bewegtbildern durch Bildverarbeitung sowie die Klassifizierung der Bewegungsrichtung. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, Januar 2004.

[Feu94] Th. Feuring. Neuronale Fuzzy-Netze – Eine Einführung. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1994. Bericht Nr. 2/94 – I.

[Feu95] Thomas Feuring. *Fuzzy-Neuronale Netze – Von kooperativen über hybride zu fusionierten vage-konnektionistischen Systemen.* PhD thesis, Westfälische Wilhelms-Universität Münster, Mathematisch-Naturwissenschaftliche Fakultät, 1995.

[FFdS95] Pablo A. Ferrari, Arnoldo Frigessi, and Paula Gonzaga de Sá. Fast approximate maximum a posteriori restoration of multicolour images. *Journal of the Royal Statistical Society – B*, 57(3):485–500, 1995.

[Fie79] R. Fiedler. Beitrag zur automatischen Untersuchung und Erkennung von Texturen in digitalisierten Bildern aus der Fernerkundung. Technical report, DFVLR – Institut für Nachrichtentechnik, Oberpfaffenhofen, 1979.

[FL95a] Th. Feuring and W. M. Lippe. Fuzzy neural networks are universal approximators. In *Proceedings of World Congress on Neural Networks*, pages 258–262, Washington, D.C., 1995.

[FL95b] Th. Feuring and W. M. Lippe. A goodness-function for trained fuzzy neural networks. In *IEEE Conference on Neural Networks*, pages 2264–2269, Perth, 1995.

[FL95c] Th. Feuring and W. M. Lippe. Über die Mächtigkeit von Fuzzy Neuronalen Netzen. In *Fuzzy-Neuro-Systeme '95: Theorie und Anwendungen, Tagungsband zum 3. Workshop*, pages 215–222, Darmstadt, 1995.

[FM81] K. S. Fu and J. K. Mui. A survey on image segmentation. In *In. Pattern Recognition*, volume Vol. 13, pages 3–16, 1981.

[FTA94] José L. Ribeiro Filho, Philip C. Treleaven, and Cesare Alippi. Genetic-algorithm programming enviroments. *IEEE Computer*, pages 28–43, June 1994.

[Fun90] Keinosuke Funkunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, San Diego, 1990.

[Gal90] L. J. Galbiati. *Maschine Vision and Digital Image Processing Fundamentals.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[Geh] Gehirn und geist. ULR of the Gehirn & Geist website, Germany. `http://www.gehirn-und-geist.de`.

[GG89] Isak Gath and Amir B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781, 1989.

[GGM84a] Pierre Goupillaud, Alex Grossmann, and Jean P. Morlet. Cycle-octave and related transforms in seismic signal analysis. In *Geoexploration*, volume 23(1), pages 85–102, 1984.

[GGM84b] Pierre Goupillaud, Alex Grossmann, and Jean P. Morlet. Cycle-octave representation for instantaneous frequency-spectra. In *Geophysics*, volume 49(5), pages 669–669, 1984.

[GHKMM87] Alex Grossmann, Matthias Holschneider, Richard Kronland-Martinet, and Jean P. Morlet. Detection of abrupt changes in sound signals with the help

of wavelet transforms. In *Advances in Electronics and Electron Physics*, volume S19, pages 289–306, 1987.

[GK79] Donald E. Gustafson and William C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proc. of the IEEE Conference on Decision and Control*, pages 761–766, San Diego, 1979.

[Gla33] Otto Glasser. *Wilhelm Conrad Röntgen and the Early History of the Roentgen Rays*. London, 1933.

[GM85] Stephen Grossberg and E. Mingolla. Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations. *Perception and Psychophysics*, 38, 1985.

[GMP85] Alex Grossmann, Jean P. Morlet, , and T. Paul. Transforms associated to square integrable group-representations: 1., general results. In *Journal of Mathematical Physics*, volume 26(10), pages 2473–2479, 1985.

[GMP86] Alex Grossmann, Jean P. Morlet, and T. Paul. Transforms associated to square integrable group-representations: 2., examples. In *Annales de l'Institut Henri Poincare – physique Theorique*, volume 45(3), pages 293–309, 1986.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading/Mass., 1989. ISBN 0-201-15767-5.

[Gol96] Alexander Goltsev. An assembly neural network for texture segmentation. *Neural Networks*, 9(4):643–653, 1996.

[Gon02] Andreu Gonzalez. *Model-based texture classification under varying illumination*. PhD thesis, Heriot-Watt University, Edinburgh, UK, 9 2002.

[GP94] Monika M. Gorkani and Rosalind W. Picard. Texture orientation for sorting photos "at a glance". In *IEEE Conference on Pattern Recognition*, Jerusalem, October 1994.

[GR89] Stephen Grossberg and M. E. Rudd. A neural network for visual motion perception: Group and element apparent motion. *Neural Networks*, 2, 1989.

[Gra95a] Amara Graps. An introduction to wavelets. In *IEEE Computational Science & Engineering*, volume 2, pages 50–61, 1995.

[Gra95b] Amara Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2(2), 1995.

[Gra95c] Adolf Grauel. *Fuzzy-Logik: Einführung in die Grundlagen mit Anwendungen.* BI Wissenschaftsverlag, Mannheim, 1995.

[Gro69a] Stephen Grossberg. Embedding fields: A theory of learning with physiological implications. *Journal of Mathematical Psychology*, 6, 1969.

[Gro69b] Stephen Grossberg. Some networks that can learn, remember, and reproduce any number of complicated space-time patterns, I. *Journal of Mathematics and Mechanics*, 19, 1969.

[Gro70] Stephen Grossberg. Some networks that can learn, remember, and reproduce any number of complicated space-time patterns, II. *Studies in Applied Mathematics*, 49, 1970.

[Gro71] Stephen Grossberg. Embedding fields: Underlying philosophy, mathematics, and applications to psychology, physiology, and anatomy. *Journal of Cybernetics*, 1, 1971.

[Gro76] Stephen Grossberg. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23, 1976.

[GS82] Madan M. Gupta and Elie Sanchez, editors. *Approximate Reasoning in Decision Analysis.* North-Holland, Amsterdam, 1982. ISBN 0-444-86492-X.

[GT03] Angelika Geyer and Sari Thiele. Segmentierung von Dot-Matrix-Zeichen und deren Klassifizierung mit Hilfe spezieller neuronaler Netze. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, Dezember 2003.

[Gup93] R. Gupta. Texture analysis of mammograms utilising the Laws matrices. Master's thesis, University of Aberdeen, Department of Biomedical Physics and Bioengineering, 1993.

[GW77] Rafael C. Gonzales and Paul Wintz. *Digital Image Processing.* Addison-Wesley, Reading, Massachusetts, 1977.

[GW92] Rafael C. Gonzales and Richard E. Woods. *Digital Image Processing.* Addison-Wesley, Reading, Massachussetts, 1992.

[GY88] Madan M. Gupta and Takeshi Yamakawa, editors. *Fuzzy Computing – Theory, Hardware, and Applications.* North-Holland, Amsterdam, 1988.

[Hab91] Peter Haberäcker. *Digitale Bildverarbeitung: Grundlagen und Anwendungen.* Hanser Verlag, München/Wien, 1991.

[Hac02] Karsten Hachmeister. Fehlererkennung in Glashohlkörpern und Klassifizierungsoptimierung mittels evolutionärer Algorithmen. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, März 2002.

[Har79] Robert M. Haralick. Statistical and structural approaches to texture. In *Proceedings of the IEEE*, volume 67/5, pages 786–804, May 1979.

[Har95a] Sonja Hartberger. Einführung in die Theorie der Fuzzy-Controller und Regeloptimierung mittels C. T. Lin und C. S. G. Lee-Netzen. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, Dezember 1995.

[Har95b] Sonja Hartberger. Radial-Basis-Function-Netze. Lehrstuhl für Informatik, WWU Münster, 1995.

[HD94] Y. Hu and T. J. Dennis. Texture image segmentation by context enhanced clustering. *IEE Proc.-Vis. Image Signal Process.*, 141(6):413–421, December 1994.

[Heb49] Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* John Wiley & Sons, New York, 1949.

[Hei02] Markus Heitmann. Texterkennung auf Gerätedisplays. Technical report, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, März 2002.

[Hei03] Markus Heitmann. Erkennung verzerrter Schriften mit Hilfe Neuronaler Netze. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, März 2003.

[HFP83] John J. Hopfield, D. I. Feinstein, and R. G. Palmer. Unlearning has a stabilizing effect in collective memories. *Nature*, 304, July 1983.

[HG94] S. K. Halgamuge and M. Glesner. Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems*, 65(1):1–12, 1994.

[HKK97] Frank Höppner, Frank Klawonn, and Rudolf Kruse. *Fuzzy-Clusteranalyse: Verfahren für die Bilderkennung, Klassifikation und Datenanalyse.* Vieweg Verlag, Braunschweig, Wiesbaden, 1997. ISBN 3-528-05543-X.

[HMM88] Paul Harmon, Rex Maus, and William Morrisey. *Expert Systems, Tools and Applications.* John Wiley and Sons, Chichester, 1988. ISBN 0-471-83950-7.

[Hoh93] Ralf Hohenstein. Klassifizierung neuromagnetischer Feldmuster mit einem Fuzzy ARTMAP-Netzwerk. Master's thesis, Institut für numerische Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1993.

[Hol92a] John H. Holland. *Adaption in Natural and Artificial Systems.* MIT Press, Cambridge/MA, 1992. ISBN 0-262-08213-6.

[Hol92b] John H. Holland. Genetische Algorithmen. *Spektrum der Wissenschaft*, 15(9):44–51, September 1992. ISSN 0170-2971.

[Hop82] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, April 1982.

[Höw94] Frank Höwing. Entwicklung einer Parallelrechnerarchitektur zur Verarbeitung eines Neuronalen Netzes. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, Februar 1994.

[HPG94] Saman K. Halgamuge, Werner Pöchmüller, and Manfred Glesner. An alternative approach for generation of membership funktions and fuzzy rules based on radial and cubic basis function networks. *International Journal of Approximate Reasoning*, 7(1), 1994.

[HS92] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume Volume I. Addison-Wesley, Reading, Massachussetts, 1992.

[HS95] W. Härdle and W. Steiger. Optimal median smoothing. *Applied Statistics*, 44(2):258–268, 1995.

[HSD73] Robert M. Haralick, K. Shanmugam, and I. Dinstein. Textual features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:610–612, 1973.

[HSZ87] R. M. Haralick, S. R. Sternberg, and X. H. Zhuang. Image analysis using mathematical morphology. *IEEE Trans. Patt. Anal. Mach. Intell.*, (PAMI-9 4):532–550, 1987.

[HT85] John J. Hopfield and David W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52, 1985.

[HT86] John J. Hopfield and David W. Tank. Computing with neural circuits: A model. *Science*, (233):625–633, 1986.

[Hun93] Chuan-Chang Hung. Building a neuro-fuzzy learning control system. *AI Expert*, pages 40–49, November 1993.

[HW96] N. Hessnielsen and M. V. Wickerhauser. Wavelets and time-frequency analysis. In *Proceedings of the IEEE*, volume 84(4), pages 523–540, 1996.

[IMKF03] D. K. Iakovidis, D. E. Maroulis, S. A. Karkanis, and I. N. Flaounas. Color texture recognition in video sequences using wavelet covariance features and support vector machines. In *Proceedings of the 29th EUROMICRO Conference*, pages 199–204, Belek-Antalya, Turkey, 9 2003.

[Jac89] Peter Jackson. *Expertensysteme – Eine Einführung.* Addison-Wesley Deutschland, Bonn, 1989. ISBN 3-89319-193-3.

[Jäh93] Bernd Jähne. *Digitale Bildverarbeitung.* Springer-Verlag, Berlin, Heidelberg, 1993.

[Jäh95] Bernd Jähne. *Digital Image Processing.* Springer-Verlag, Berlin, Heidelberg, 1995.

[Jan94] Anne Jankrift. Klassifizierung neuromagnetischer Feldmuster unter Verwendung eines Cascade-Correlation Netzes. Master's thesis, Institut für numerische Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1994.

[Jar85] J. P. Jaroslavskij. *Einführung in die digitale Bildverarbeitung.* VEB Deutscher Verlag der Wissenschaften, Berlin, 1985.

[Jar90] H. Jaroslavskij. *Erfassung und maschinelle Verarbeitung von Bilddaten.* Hütig Buch Verlag, Heidelberg, 1990.

[Jas97] Brigitte Jaschke. *Glasherstellung: Produkte, Technik, Organisation.* Deutsches Museum, München, 1997.

[Jav] JavaNNS. ULR of the Java version of the Stuttgarter Neuronale Netze Simulator at the University of Tübingen. `http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome.html`.

[JMNS96] Bernd Jähne, Robert Massen, Bertram Nickolay, and Harald Scharfenberg. *Technische Bildverarbeitung – Maschinelles Sehen.* Springer-Verlag, Berlin, Heidelberg, 1996.

[Jon85] Kenneth De Jong. Genetic algorithms: A 10 year perspective. In *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, pages 169–177, 1985.

[JS94] B. Jawerth and W. Sweldens. An overview of wavelet-based multiresolution analysis. In *SIAM Review*, volume 36(3), pages 377–412, 1994.

[Kai94] Gerald Kaiser. *A Friendly Guide to Wavelets.* Birkhäuser, Boston, 1994.

[Kar91] C. L. Karr. Genetic algorithms for fuzzy logic controllers. *AI Expert*, 6(2):26–33, 1991. ISSN 0888-3785.

[Kel94] Patrick M. Kelly. An algorithm for merging hyperellipsodial clusters. Computer research group technical report, Los Alamos National Laboratory, 1994.

[KF93] Jörg Kahlert and Hubert Frank. *Fuzzy-Logik und Fuzzy-Control.* Vieweg-Verlag, Braunschweig/Wiesbaden, 1993. ISBN 3-528-05304-6.

[KGK93] Rudolf Kruse, Jörg Gebhardt, and Frank Klawonn. *Fuzzy-Systeme.* Leitfäden und Monographien der Informatik. Teubner-Verlag, Stuttgart, 1993.

[KGP94] Rudolf Kruse, Jörg Gebhardt, and Rainer Palm, editors. *Fuzzy-Systems in Computer Science.* Artificial Intelligence/Künstliche Intelligenz. Vieweg Verlag, Wiesbaden, 1994.

[KHW92] Patrick M. Kelly, Don R. Hush, and James M. White. An adaptive algorithm for modifying hyperellipsodial decision surfaces. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 196–201, 1992.

[KK96] Frank Klawonn and Rudolf Kruse. Derivation of fuzzy classification rules from multidimensional data. Fachbereich Informatik, Universität Braunschweig, 1996.

[KKL92] Teuvo Kohonen, Jari Kangas, and Jorma Laaksonen. SOM_PAK – the Self-Organizing Map program package, version 1.2. Technical report, Laboratory of Computer and Information Science, Helsinki University of Technology, November 1992.

[KKLT92] Teuvo Kohonen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK – the Learning Vector Quantization program package, version 2.1. Technical report, Laboratory of Computer and Information Science, Helsinki University of Technology, October 1992.

[Kle96] Carsten Klevenhusen. Texture analysis with artificial neural networks: Texture feature extraction using the wavelet transform. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel / University of Glamorgan, Wolfenbüttel, Pontypridd, October 1996.

[Kli95] Adelheid Klingebiel. Fuzzy-Neuronale Netze nach Sankar K. Pal und Sushmita Mitra. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, November 1995.

[Koh82] Teuvo Kohonen. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[Koh84] Teuvo Kohonen. *Self-organization and associative memory*, volume 8 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin/Heidelberg/New York, 1984.

[Koh88] Teuvo Kohonen. *An Introduction to Neural Computing*. Pergamon Press, New York, 1988. pp. 3–16.

[Koh89] Teuvo Kohonen. *Self-Organisation and Associative Memory*. Number 8 in Springer Series Information Science. Springer Verlag, Berlin/Heidelberg, 1989. ISBN 3-540-51387-6.

[Köh90a] Monika Köhle. *Neuronale Netze*. Springers Angewandte Informatik. Springer Verlag, Wien/New York, 1990. ISBN 3-211-82220-8.

[Koh90b] Teuvo Kohonen. The self-organizing map. *Proceedings of IEEE*, 78:1464–1480, 1990.

[Kos92] Bart Kosko. *Neural Networks and Fuzzy Systems*. Prentice Hall, Englewood Cliffs, New Jersey, 1992. ISBN 13-612334-1.

[Kow95] Michael Kowatz. ART Adaptive Resonance Theory – Aufmerksamkeitsgesteuerte Systeme. Lehrstuhl für Informatik, WWU Münster, 1995.

[KP96] Vassili Kovalev and Maria Petrou. Multidimensional co-occurrence matrices for object recognition and matching. *Graphical Models and Image Processing*, 58(3):187–197, May 1996.

[KR90] Leonard Kaufmann and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York, 1990.

[Kri93] R. Krishnapuram. Fuzzy clustering methods in computer vision. In *Proceedings of the 1st European Congress on Intelligent Techniques and Soft Computing*, volume 2, pages 720–730, Aachen, 1993.

[KS93] Erich Peter Klement and Wolfgang Slany, editors. *Fuzzy Logic in Artificial Intelligence – 8th Austrian Artificial Intelligence Conference, FLAI'93*, volume 695 of *Lecture Notes in Artificial Intelligence*, Linz, June 1993. Springer-Verlag, Berlin/Heidelberg.

[KTBC95] T. Kasparis, N. S. Tzannes, M. Bassiouni, and Q. Chen. Texture description using fractal and energy features. *Computers & Electrical Engineering*, 21(1):21–32, 1995.

[Küh95] Sven Kühne. Neuronale Netze und Genetische Algorithmen – Ein Überblick über aktuelle Ansätze. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1995.

[Küh96] Sven Kühne. Universalitätsbeweise. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, Januar 1996.

[Kur89] Karl Kurbel. *Entwicklung und Einsatz von Expertensystemen – Eine anwendungsorientierte Einführung in wissensbasierte Systeme.* Springer Verlag, Berlin/Heidelberg, 1989. ISBN 3-540-51013-3.

[Law80a] Kenneth Ivan Laws. Rapid texture identification. *SPIE Image Processing for Missile Guidance*, 238:376–380, 1980.

[Law80b] Kenneth Ivan Laws. *Textured Image Segmentation.* PhD thesis, University of Southern California, January 1980.

[LCC95] L. M. Linnett, D. R. Carmichael, and S. J. Clarke. Texture classification using a spatial-point process model. *IEE Proc.-Vis. Image Signal Process.*, 142(1):1–6, February 1995.

[LE89] Hansl Loos and Roland Golpon (Ed.). *Farbmessung.* Beruf + Schule, Itzehoe, 1989.

[Lee80] Jong-Sen Lee. Digital image enhancement and noise-filtering by use of local statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(2):165–168, March 1980.

[Lei95] Donald Dewar Leitch. *A New Genetic Algorithm for the Evolution of Fuzzy Systems.* PhD thesis, University of Oxford, Department of Engineering Science, Robotics Research Group, 1995.

[LF93] Andrew Laine and Jian Fan. An adaptive approach for texture segmentation by multi-channel wavelet frames. Center for Computer Vision and Visualization, August 1993.

[LFB95] W. M. Lippe, Th. Feuring, and Th. Büscher. A fuzzy-neural network based on the backpropagation algorithm. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1995. Bericht Nr. 10/95 – I.

[LFH94] W. M. Lippe, Th. Feuring, and R. Hohenstein. Klassifizierung neuromagnetischer Feldmuster mit einem Fuzzy ARTMAP-Netzwerk. Technical report, Institut für numerische Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1994. Bericht Nr. 1/94 – I.

[LFJ94] W. M. Lippe, Th. Feuring, and A. Jankrift. Klassifizierung neuromagnetischer Feldmuster unter Verwendung eines Cascade-Correlation Netzes. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1994. Bericht Nr. 3/94 – I.

[LFJH94a] W. M. Lippe, Th. Feuring, A. C. Jankrift, and R. Hohenstein. Frequency-domain localization of intracerebral dipole sources. In C. H. Dagli, B. R. Fernandez, J. Gosh, and R. T. Soundra Kumara, editors, *Proceedings of International Conference on Artificial Neural Networks in Engineering*, Intelligent Engineering Systems through Artificial Neural Networks, pages 747–752, San Louis, 1994. ASME Press, New York.

[LFJH94b] W. M. Lippe, Th. Feuring, A. C. Jankrift, and R. Hohenstein. Klassifikation neuromagnetischer Daten mittels künstlicher Neuronaler Netze. In *Mustererkennung 1994*, pages 548–556. Springer-Verlag, Wien, 1994.

[LFM95] W. M. Lippe, Th. Feuring, and L. Mischke. Supervised learning in fuzzy neural network. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1995. Bericht Nr. 12/95 - I.

[LFT94] W. M. Lippe, Th. Feuring, and A. Tenhagen. A fuzzy-controlled delta-bar-delta learningrule. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1686–1690, Orlando, Florida, 1994.

[LH89] G. E. Liepins and M. R. Hilliard. Genetic algorithms: Foundations and applications. *Annals of Operations Research*, 21:31–58, 1989.

[LMR94] A. K. Louis, P. Maas, and A. Rieder. *Wavelets.* B. G. Teubner, Stuttgart, 1994.

[Lóp03] Nuria Garrido López. Fehlererkennung bei Navigationsgeräte-Displays. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, September 2003.

[LSdWD97] S. Livens, P. Scheunders, G. Van de Wouwer, and D. Van Dyck. Wavelets for texture analysis. Web-report, VisieLab, Department of Physics, University of Antwerpen, 1997.

[LST04] S. Liapis, E. Sifakis, and G. Tziritas. Colour and texture segmentation using wavelet frame analysis, deterministic relaxation, and fast marching algorithms. *Journal of Visual Communication and Image Representation*, 15(1):1–26, 3 2004.

[LT04] S. Liapis and G. Tziritas. Colour and texture image retrieval using chromaticity histograms and wavelet frames. *IEEE Transactions on Multimedia*, 6(5):676–686, 10 2004.

[LXH98] Guoqing Liu, Hong Xiong, and Shunji Huang. Multiresolution analysis and classification of the textured SAR image with wavelet transform. In Ji Zhou

and Anil K. Jain and, editors, *International Symposium on Multispectral Image Processing (ISMIP'98)*, volume 3545, pages 173–176, Bellingham, Washington, 1998. SPIE–The International Society for Optical Engineering. ISBN 0-8194-3006-4.

[Mac67] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium*, volume 1, pages 281–297, Berkeley, California, 1967.

[Mäe03] Topi Mäenpää. *The local binary pattern approach to texture analysis -- extensions and applications.* PhD thesis, University of Oulu, Oulu, Finland, 2003.

[Mal89] Stéphane Mallat. A theory for multiresolution signal decomposition: the wavelet representation. In *IEEE, Trans. on Pattern Anal. Machine Intell.*, volume 11(7), pages 674–693, 1989.

[Mal91] Stéphane Mallat. Zero crossing of a wavelet transform. In *IEEE Trans. Inform. Theory*, volume 37, pages 1019–1033, 1991.

[Mal97] Stefan Malon. Entwicklung eines Systems zur Texturanalyse mittels Wavelet-Transformation und neuronalem Netz für das Bildverarbeitungssystem WiT. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, November 1997.

[Mat04] Dario Matos. Erkennung von Fahrbahnrändern in Bewegtbildern durch Bildverarbeitung. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, Januar 2004.

[McQ87] Louis L. McQuitty. *Pattern-Analytic Clustering: Theory, Method, Research, and Configural Findings.* University Press of America, Lanham, NY, 1987.

[MGF02] Mauricio Pozzobon Martins, Lamartine N. Frutuoso Guimaratilde, and Leila Maria Garcia Fonseca. Texture feature neural classifier for remote sensing image retrieval systems. In *XV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'02)*, 2002.

[MGTW94] W. P. J. Mackeown, P. Greenway, B. T. Thomas, and W. A. Wright. Contextual image labelling with a neural network. *IEE Proc.-Vis. Image Signal Process.*, 141(4):238–244, August 1994.

[MH95] S. Marriott and R. Harrison. A modified fuzzy ARTMAP architecture for the approximation of noisy mappings. *Neural Networks*, 8(4):619–641, 1995.

[MMSW93] Andreas Mayer, Bernhard Melcher, Andreas Schildwein, and Rainer Wolke. *Fuzzy Logic – Einführung und Leitfaden zur praktischen Anwendung – Mit Fuzzy-Shell in C++.* Addison-Wesley (Deutschland), Bonn, 1993. ISBN 3-89319-443-6.

[Mor81] Jean P. Morlet. Sampling theory and wave propagation. In *Proc. 51st. Annu. Meet. Soc. Explor. Geophys.*, Los Angeles, 1981.

[MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MP69] Marvin L. Minsky and Seymour S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA, 1969.

[MP04] Topi Mäenpää and Matti Pietikäinen. Classfication with color and texture: jointly or separately? *Pattern Recognition*, 37(8):1629–1640, 2004.

[MR95] W. B. Mikhael and A. Ramaswamy. Resolving images in multiple transform domains with applications. *Digital Signal Processing*, 5:81–90, 1995.

[MTP03] Topi Mäenpää, Markus Turtinen, and Matti Pietikäinen. Real-time surface inspection by texture. *Real-Time Imaging*, 9(5):289–296, 2003.

[Mül90a] Reinert H. G. Müller. Die Fouriertransformation in der Bildverarbeitung – 1. Teil. *Elektronik*, 3:50–59, 1990.

[Mül90b] Reinert H. G. Müller. Die Fouriertransformation in der Bildverarbeitung – 2. Teil. *Elektronik*, 4:72–80, 1990.

[Neu96a] NeuroFuzzy computing, 1996. NIBS Pte Ltd Technical Report TR-960308.

[Neu96b] NeuroGenetic computing, 1996. NIBS Pte Ltd Technical Report TR-960309.

[Neu96c] Stefan Neuwirth. Evolutionäre Strukturierung von Neuronalen Netzen. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, Mai 1996.

[Neu97] Stefan Neuwirth. Kombination Neuronaler Netze mit Genetischen Algorithmen. Lehrstuhl für Informatik, WWU Münster, 1997.

[Nie82] Klaus Niederdrenk. *Die endliche Fourier- und Walsh-Transformation mit einer Einführung in die Bildverarbeitung.* Friedrich Vieweg & Sohn, Braunschweig/Wiesbaden, 1982.

[Nie83] H. Niemann. *Klassifikation von Mustern.* Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1983.

[NK95] Detlef Nauck and Rudolf Kruse. NEFCLASS – a neuro-fuzzy approach for the classification of data. In K. M. George, Janice H. Carrol, Ed Dean, Dave Oppenheim, and Jim Hightower, editors, *Proceedings of the 1995 ACM Symposium on Applied Computing*, Applied Computing 1995, Nashville, 1995. ACM Press.

[NK96] Detlef Nauck and Frank Klawonn. Neuro-fuzzy classification initialized by fuzzy clustering. In *Proc. Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT96)*, 1996.

[NKK93] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. Combining neural networks and fuzzy controllers. In Erich Peter Klement and Wolfgang Slany, editors, *Fuzzy Logic in Artificial Intelligence - 8th Austrian Artificial Intelligence Conference, FLAI'93*, volume 695 of *Lecture Notes in Artificial Intelligence*, pages 35–46, Linz, June 1993. Springer-Verlag, Berlin/Heidelberg.

[NKK94] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Neuronale Netze und Fuzzy-Systeme.* Künstliche Intelligenz. Vieweg Verlag, Braunschweig/Wiesbaden, 1994.

[NNK96] Detlef Nauck, Ulrike Nauck, and Rudolf Kruse. Generating classification rules with the neuro-fuzzy system NEFCLASS. In *Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS'96)*, Berkeley, 1996.

[Nor92] T. Norita. Engineering applications of fuzzy systems – applications of image processing and understanding with fuzzy theory. In *Joint Japanese-European Symposium on Fuzzy Systems Berlin, Document 17*, 1992.

[NS95] G. P. Nason and B. W. Silverman. The stationary wavelet transform and some statistical applications. Department of Mathematics, University of Bristol, UK, 1995.

[OPN96] Timo Ojala, Matti Pietikäinen, and Jarkko Nisula. Determining composition of grain mixtures by texture classification based on feature distributions. *International Journal of Pattern Recognition and Artificial Intelligence*, 10:73–82, 1996.

[OQ94] Finbarr O'Sullivan and Maijian Qian. A regularized contrast statistic for object boundary estimation—implementation and statistical evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):561–570, June 1994.

[Pao89] Yoh-Han Pao. *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley, Reading/Mass., 1989. ISBN 0-201-12584-6.

[Pav90] Theo Pavlidis. *Algorithmen zur Grafik und Bildverarbeitung.* Heise Verlag, Hannover, 1990.

[PFJ03] H. Permuter, J. Francos, and I. H. Jermyn. Gaussian mixture models of texture and colour for image database retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2003)*, Hong Kong, China, 4 2003.

[Pic92a] Rosalind W. Picard. Random field texture coding. In *Society for Information Display 1992 International Symposium Digest*, volume XXIII, pages 685–688, Boston, May 1992.

[Pic92b] Rosalind W. Picard. Structured patterns for random fields. In *Proceedings of the 26th Annual Asilomar Conference on Signals, Systems, and Computers*, pages 1011–1015, October 1992.

[Pic95] Rosalind W. Picard. Digital libraries: Meeting place for high-level and low-level vision. In *Asian Conference on Computer Vision*, Singapore, December 1995.

[PKL93] Rosalind W. Picard, Tanweer Kabir, and Fang Liu. Real-time recognition with the entire Brodatz texture database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 638–639, June 1993.

[PKL94] Daihee Park, Abraham Kandel, and Gideon Langholz. Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):39–47, January 1994.

[PL00] Rupert Paget and I. Dennis Longstaff. Open-ended texture classification for terrain mapping. In *IEEE International Conference on Image Processing (ICIP'2000)*, volume 3, pages 584–587, Vancouver, BC, Canada, 9 2000.

[PM94] R. W. Picard and T. P. Minka. Vision texture for annotation. Technical report, MIT Media Laboratory Perceptual Computing Group, 1994? Technical Report #302.

[PNMT04] Matti Pietikäinen, Tomi Nurmela, Topi Mäenpää, and Markus Turtinen. View-based recognition of real-world textures. *Pattern Recognition*, 37(2):313–323, 2004.

[PP91] Rosalind W. Picard and Alex P. Pentland. Markov/Gibbs modeling: Texture and temperature. In *Boston*, November 1991.

[PP93] Kris Popat and Rosalind W. Picard. Novel cluster based probability model for texture synthesis, classification, and compression. In *SPIE Visual Communications and Image Processing '93*, Boston, November 1993.

[PP94] Kris Popat and Rosalind W. Picard. Cluster-based probability model applied to image restoration and compression. Perceptual Computing Group Technical Report 253, MIT Media Laboratory, 1994.

[QN93] Mohamed Quafafou and Mohammed Nafia. GAITS: Fuzzy sets-based algorithms for computing strategies using genetic algorithms. In Erich Peter Klement and Wolfgang Slany, editors, *Fuzzy Logic in Artificial Intelligence – 8th Austrian Artificial Intelligence Conference, FLAI'93*, volume 695 of *Lecture Notes in Artificial Intelligence*, pages 59–67, Linz, June 1993. Springer-Verlag, Berlin/Heidelberg.

[Ran97] Trygve Randen. *Filter and Filter Bank Design for Image Texture Recognition.* PhD thesis, Norwegian University of Science and Technology, Stavanger, Norway, 1997.

[Rei96] L.-M. Reissell. Wavelet multiresolution representation of curves and surfaces. *Graphical Models and Image Processing*, 58(3):198–217, May 1996.

[RG96] Michael Rath and Christof Gemke. Neuronale Netze – Adaptive Resonance Theory (ART). Lehrstuhl für Informatik, WWU Münster, 1996.

[Ric81] Manfred Richter. *Einführung in die Farbmetrik.* Walter de Gruyter, 1981.

[Ric93] John A. Richards. *Remote Sensing Digital Image Analysis.* Springer-Verlag, Berlin, 1993.

[RMt86] David E. Rumelhart, James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing – Explorations in the Microstructure of Cognition*, volume 1 – Foundations. MIT Press, Cambridge, Massachusetts, 1986.

[Roc] Roche. ULR of the Roche Lexikon Medizin at gesundheit.de, Germany. `http://www.gesundheit.de/roche/`.

[Roj93] Raúl Rojas. *Theorie der neuronalen Netze.* Springer-Verlag, Berlin, 1993.

[Rom84] H. Charles Romesburg. *Cluster Analysis for Researchers.* Lifetime Learning Publications, Belmont, CA, 1984.

[Rom94] Heinrich Rommelfanger. *Fuzzy Decision Support-Systeme – Entscheiden bei Unschärfe.* Hochschultexte. Springer-Verlag, Berlin, 1994.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 1958.

[RP00] S. Rajasekaran and G. A. Vijayalakshmi Pai. Image recognition using simplified fuzzy ARTMAP augmented with a moment based feature extractor. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(8):1081–1095, 2000.

[RZC+98] Xianyi Ren, Guilin Zhang, Zhaoyang Chen, Ruolan Hu, and Xiaohui Li. Texture segmentation based on wavelet transform. In Ji Zhou and Anil K. Jain and, editors, *International Symposium on Multispectral Image Processing (ISMIP'98)*, volume 3545, pages 432–435, Bellingham, Washington, 1998. SPIE–The International Society for Optical Engineering. ISBN 0-8194-3006-4.

[SA01a] Alexander Stolpmann and Jürgen Angele. Erkennung und Bewertung von Fehlerstrukturen in maschinell gefertigtem durchsichtigem Hohlglas. In *Innovationsland Niedersachsen, Wissenschaft und Wirtschaft auf der Hannover Messe 2001*, 4 2001.

[SA01b] Alexander Stolpmann and Jürgen Angele. Erkennung und Bewertung von Fehlerstrukturen in maschinell gefertigtem durchsichtigem Hohlglas. In *Science-Report*, volume 1. Fachhochschule Braunschweig/Wolfenbüttel, 11 2001.

[SA01c] Alexander Stolpmann and Jürgen Angele. Erkennung und Bewertung von Fehlerstrukturen in maschinell gefertigtem durchsichtigem Hohlglas. Technical report, Fachhochschule Braunschweig/Wolfenbüttel, 2001.

[SA04] Golam Sorwar and Ajith Abraham. Dct based texture classification using soft computing approach. *Malaysian Journal of Computer Science*, 17(1), 2004.

[SAD00a] Alexander Stolpmann, Jürgen Angele, and Laurence S. Dooley. Quality inspection of veneer using soft-computing methods. In *Evolutionary Design and Manufacture*, pages 363–370. Springer-Verlag, London, 2000.

[SAD00b] Alexander Stolpmann, Jürgen Angele, and Laurence S. Dooley. Soft-Computing System zur Fehlererkennung in Furnierholz. In *Symposium, Anwendung von Fuzzy Technologien und Neuronalen Netzen*, Clausthal-Zellerfeld, 3 2000.

[SAD00c] Alexander Stolpmann, Jürgen Angele, and Laurence S. Dooley. Soft-computing System zur Fehlererkennung in Furnierholz. In *AFN-Berichte 2000*, pages 99–107. Clausthal-Zellerfeld, 2000.

[Sag85] Gerhard Sagerer. *Darstellung und Nutzung von Expertenwissen für ein Bildanalysesystem.* Springer Verlag, Heidelberg, Berlin, 1985.

[SCBW94] Robert N. Sharpe, Mo-yuen Chow, Steve Briggs, and Larry Windingland. A methodology using fuzzy logic to optimize feedforward artificial neural network configurations. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(5):760–768, May 1994.

[Sch88] Joachim Schwarz. *Digitale Verarbeitung stochastischer Signale.* Oldenbourg Verlag, München/Wien, 1988.

[Sch93a] Eberhard Schönburg, editor. *Industrielle Anwendung Neuronaler Netze.* Addison-Wesley, Bonn, 1993.

[Sch93b] Ulrich Schulte. *Einführung in Fuzzy Logik – Fortschritt durch Unschärfe.* Franzis-Verlag, München, 1993. ISBN 3-7723-4771-1.

[SD98] Alexander Stolpmann and Laurence S. Dooley. Genetic algorithms for automised feature selection in a texture classification system. In *Proceedings of the 4^{th} International Conference on Signal Processing*, Beijing, 1998.

[SD99a] Alexander Stolpmann and Laurence S. Dooley. About the use of fuzzy clustering for texture classification. In *Proceedings of the 18^{th} International Conference of the North American Fuzzy and Information Processing Society, NAFIPS'99*, New York, 1999.

[SD99b] Alexander Stolpmann and Laurence S. Dooley. A texture classification system using statistical and soft-computing methods. In *Proceedings of the 1st Infotech Oulu Workshop on Texture Analysis in Machine Vision*, Oulu, Finland, 1999.

[SD99c] Alexander Stolpmann and Laurence S. Dooley. A texture classification system with automatic feature vector optimization using genetic algorithms. In *Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing, EUFIT'99*, Aachen, 1999.

[SD00] Alexander Stolpmann and Laurence S. Dooley. A texture classification system using statistical and soft-computing methods. In *Texture Analysis Book*, Machine Perception and Artificial Intelligence (MPAI) series. World Scientific, 2000.

[Sen04] Christian Senft. Automatische Skriptverwaltung für neuronale Netze. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, Juni 2004.

[Sha48] Claude E. Shannon. A mathematical theory of communication. In *Bell System Technical Journal*, pages 27:379–423 and 623–656, July and October 1948.

[Sha93] Prathap S. Shanmugam. Co-dependency matrix and its implementation and installation under the KHOROS programming enviroment. Department of Electrical Engineering, Course ECE 599, Technical Report, 1993.

[SHB93] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis and Machine Vision.* Chapman & Hall Computing, London, 1993.

[SHF94] Eberhard Schöneburg, Frank Heinzmann, and Sven Feddersen. *Genetische Algorithmen und Evolutionsstrategien.* Addison-Wesley (Deutschland), Bonn, 1994. ISBN 3-89319-493-2.

[SHG90] Eberhard Schönburg, Nikolaus Hansen, and Andreas Gawelczyk. *Neuronale Netzwerke.* Markt & Technik Verlag, Haar bei München, 1990.

[SL91] Yi Shang and Guo-Jie Li. New crossover operators in genetic algorithms. In *Proceedings of the 1991 IEEE International Conference on Tools for AI*, pages 150–153, San Jose, California, 1991.

[SM02] Arne Schäffler and Nicole Menche. *Lehrbuch und Atlas des menschlichen Körpers: Anatomie, Physiologie, Krankheitsbilder.* Komet-Verlag, Frechen, 2002. ISBN 3-89836-225-6.

[SMM+91] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A comparison of genetic sequencing operators. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, 1991.

[SMM94] B. G. Sherlock, D. M. Monro, and K. Millard. Fingerprint enhancement by directional Fourier filtering. *IEE Proc.-Vis. Image Signal Process.*, 141(2):87–94, April 1994.

[SNN] SNNS. ULR of the Stuttgarter Neuronale Netze Simulator at the University of Tübingen. `http://www-ra.informatik.uni-tuebingen.de/SNNS/`.

[Soi98] Pierre Soille. *Morphologische Bildverarbeitung: Grundlagen, Methoden, Anwendungen.* Springer, Berlin, Heidelberg, New York, 1998. ISBN 3-540-64323-0.

[Sol89] Hans-Jürgen Soll. *KI Expertensysteme programmieren.* Franzis-Verlag, München, 1989.

[SP94] M. Srinivas and Lalit M. Patnaik. Genetic algorithms: A survey. *IEEE Computer*, pages 17–26, June 1994.

[SS03] Alexander Stolpmann and Wolfgang Schneider. Erkennung und Bewertung von stark verzerrten, verschmutzten oder fehlstellenbehafteten maschinell gedruckten Schriften. Technical report, Fachhochschule Braunschweig/Wolfenbüttel, 2003.

[SS04a] Alexander Stolpmann and Wolfgang Schneider. Automatisches Lesen von verzerrten und verschmierten Aufdrucken mit Hilfe neuronaler Netze. In *Innovationsland Niedersachsen, Wissenschaft und Wirtschaft auf der Hannover Messe 2004*, 4 2004.

[SS04b] Alexander Stolpmann and Wolfgang Schneider. Automatisches Lesen von verzerrten und verschmierten Aufdrucken mit Hilfe neuronaler Netze. Technical report, ti – Technologie-Informationen niedersächsischer Hochschulen, 4 2004.

[Ste93a] Roger Stein. Preprocessing data for neural networks. *AI Expert*, pages 32–37, March 1993.

[Ste93b] Rainer Steinbrecher. *Bildverarbeitung in der Praxis.* Oldenbourg Verlag, München/Wien, 1993.

[STG96] Harold Szu, Brian Telfer, and Joseph Garcia. Wavelet transform and neural networks for compression and recognition. *Neural Networks*, 9(4):695–708, 1996.

[Sto93] Alexander Stolpmann. Fuzzy Logik: Eine Einführung ins Unscharfe. Master's thesis, Fachhochschule Braunschweig/Wolfenbüttel, Wolfenbüttel, August 1993.

[Sto96] Alexander Stolpmann. Texture classification with statistical and fuzzy-clustering methods. In *Proceedings of ISDSP'96*, London, 1996.

[Sto97] Alexander Stolpmann. *Automatic Recognition of Coarse-Grained Non-Regular Structures using Non-Parametric Processing and Soft-Computing Methods – A Texture Classification System.* Pontypridd, 1997. MPhil/PhD Transfer Document.

[Sto04] Alexander Stolpmann. Einsatz neuronaler Netze bei der Erkennung unsauber gedruckter Schriften. In *Information-Mining und Wissensmanagement in Wissenschaft und Wirtschaft*, volume 7 of *Göttinger Symposium Soft Computing*, pages 73–92. Göttingen, 2004.

[Str89] Gilbert Strang. Wavelets and dilation equations: a brief introduction. In *Siam Review*, volume 31(4), pages 614–627, 1989.

[SV95] K. Shanmukh and Y. V. Venkatesh. Generalised scheme for optimal learning in recurrent neural networks. *IEE Proc.-Vis. Image Signal Process.*, 142(2):71–77, April 1995.

[Tat87] S. Tatari. *Auswahl und von Texturanalyseverfahren zur Automatisierung industrieller Sichtprüfungen.* VDI Verlag, Düsseldorf, 1987.

[Til91] Thomas Tilli. *Fuzzy-Logik – Grundlagen, Anwendungen, Hard- und Software.* Franzis-Verlag, München, 1991. ISBN 3-7723-4321-X.

[Til92] Thomas Tilli. *Automatisierung mit Fuzzy-Logik.* Franzis-Verlag, München, 1992. ISBN 3-7723-4411-9.

[Til93] Thomas Tilli. *Mustererkennung mit Fuzzy-Logik: Analysieren, klassifizieren, erkennen und diagnostizieren.* Franzis-Verlag, München, 1993. ISBN 3-7723-4871-8.

[Tiz98] Hamid R. Tizhoosh. *Fuzzy-Bildverarbeitung: Einführung in Theorie und Praxis.* Springer, Berlin, Heidelberg, New York, 1998. ISBN 3-540-63137-2.

[TK94] T. S. C. Tan and J. Kittler. Colour texture analysis using colour histogram. *IEE Proc.-Vis. Image Signal Process.*, 141(6):403–412, December 1994.

[TL93] Hideyuki Takagi and Michael Lee. Neural networks and genetic algorithm approaches to auto-design of fuzzy systems. In Erich Peter Klement and Wolfgang Slany, editors, *Fuzzy Logic in Artificial Intelligence – 8th Austrian Artificial Intelligence Conference, FLAI'93*, volume 695 of *Lecture Notes in Artificial Intelligence*, pages 68–79, Linz, June 1993. Springer-Verlag, Berlin/Heidelberg.

[TL94] R. I. Taylor and P. H. Lewis. 2D shape signature based on fractal measurement. *IEE Proc.-Vis. Image Signal Process.*, 141(6):422–430, December 1994.

[Tra93] Dirk H. Traeger. *Einführung in die Fuzzy-Logik.* B. G. Teubner, Stuttgart, 1993. ISBN 3-519-06162-7.

[Try39] Robert C. Tryon. *Cluster Analysis.* Edward Bros., Ann Arbor, 1939.

[US96] Jayaram K. Udupa and Supun Samarasekera. Fuzzy connectedness and object definition: Theory, algorithms, and application in image segmentation. *Graphical Models and Image Processing*, 58(3):246–261, May 1996.

[USC] Usc. ULR of the USC Signal and Image Processing Institute, provider of the Brodatz textures. `http://sipi.usc.edu/services/database/Database.html`.

[UWL] Uwlax. ULR of the University of Wisconsin – La Crosse. `http://www.uwlax.edu/StudentHealth/`.

[vdEPV93] Petra A. van den Elsen, E. J. D. Pol, , and Max A. Viergever. Medical image matching – a review with classification. In *In IEEE Engineering in Medicine and Biology*, number (12:1), pages 26–39, 1993.

[vdEV90] Ad W. M. van den Enden and Niek A. M. Verhoeckx. *Digitale Signalverarbeitung.* Vieweg Verlag, Braunschweig, 1990.

[vdM73] C. von der Malsburg. Self-organisation of orientation sensitive cells in the striate cortex. *Kybernetik*, (14):85–100, 1973.

[Vis] Vision texture. ULR of Vision and Modeling Group at the MIT Media Laboratory. `http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html`.

[VMC94] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proc.-Vis. Image Signal Process.*, 141(4):245–250, August 1994.

[vob] vobs. ULR of the Vorarlberger Bildungsserver, Austria. `http://www.vobs.at/bio/a-phys`.

[vV95] Astrid Dominique von Viebahn. Fuzzy-Neuronale Netze nach Y. Hayashi und H. Ishibuchi. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, Dezember 1995.

[Wal95] Matthias Walter. Einführung in die Evolutionäre Berechnung. Technical report, Institut für numerische und instrumentelle Mathematik/Informatik, Westfälische Wilhelms-Universität Münster, 1995.

[Wat86] Donald A. Waterman. *A Guide to Expert Systems.* Addison-Wesley, Reading/Mass., 1986. ISBN 0-201-08313-2.

[Web93] Webster's new encyclopedic dictionary. Black Dog & Leventhal Publishers, New York, 1993.

[Wei02] Barbara Weitz. *Atlas der Anatomie: Organsysteme und Strukturen.* Komet-Verlag, Frechen, 2002. ISBN 3-89836-234-5.

[Wer85] Diederich Wermser. *Automatische Texturauswertung auf der Basis einer Analyse des visuellen Systems*, volume 46 of *Reihe 10: Angewandte Informatik.* VDI Verlag, Düsseldorf, 1985.

[Wes96] Ralph Wesselmann. Grenzwertuntersuchung im Fuzzy-c-Means Algorithmus. Lehrstuhl für Informatik, WWU Münster, 1996.

[Wig92] Ralphe Wiggins. Docking a truck: A genetic fuzzy approach. *AI Expert*, pages 28–35, May 1992.

[Wik05] Definition of texture. ULR of Wikipedia, the free encyclopedia., 2005. `http://en.wikipedia.org/wiki/Texture/`.

[WRW94] G. Wade, A. Roberts, and G. Williams. Multiplier-less FIR filter design using a genetic algorithm. *IEE Proc.-Vis. Image Signal Process.*, 141(3):175–180, June 1994.

[WSS94] C. J. Wu, A. H. Sung, and H. S. Soliman. A fuzzy ART network with fuzzy control for image data compression. In *Proc. of IASTED Int. Conf. on Modeling, Simulation and Control in the Process Industry*, pages 95–98, May 1994.

[Wu03] Jiahua Wu. *Rotation Invariant Classification of 3D Surface Texture Using Photometric Stereo.* PhD thesis, Heriot-Watt University, Edinburgh, UK, 2003.

[YL94] F. T. S. Yu and G. W. Lu. Short-time fourier-transform and wavelet transform with fourier-domain processing. In *Applied Optics*, volume 33(23), pages 5262–5270, 1994.

[Zad65] Lotfi Asker Zadeh. Fuzzy sets. *Information and Control*, 9(8):338–353, August 1965. ISSN 0019-9958.

[Zad73] Lotfi Asker Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:28–44, 1973.

[Zad75a] Lotfi Asker Zadeh. The concept of a linguistic variable and its application to approximate reasoning (part 1). *Information Sciences*, pages 199–249, 1975.

[Zad75b] Lotfi Asker Zadeh. The concept of a linguistic variable and its application to approximate reasoning (part 2). *Information Sciences*, pages 301–357, 1975.

[Zad76a] Lotfi Asker Zadeh. The concept of a linguistic variable and its application to approximate reasoning (part 3). *Information Sciences*, pages 43–80, 1976.

[Zad76b] Lotfi Asker Zadeh. A fuzzy approach to the definition of complex or imprecise concepts. *International Journal on Man-Machine Studies*, 8:249–291, 1976.

[Zad82] Lotfi Asker Zadeh. A note on prototype theory. *Cognition*, 12:291–297, 1982.

[Zad89] Lotfi Asker Zadeh. Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):89–100, March 1989.

[Zad93a] Lotfi Asker Zadeh. Mit "Softcomputing" zur meßbaren Maschinen-Intelligenz. *Elektronik*, Sonderheft Elektronik Plus(2/1993):6, 1993. ISSN 0943-6510.

[Zad93b] Lotfi Asker Zadeh. The role of fuzzy logic and soft computing in the conception and design of intelligent systems. In Erich Peter Klement and Wolfgang Slany, editors, *Fuzzy Logic in Artificial Intelligence – 8th Austrian Artificial Intelligence Conference, FLAI'93*, volume 695 of *Lecture Notes in Artificial Intelligence*, page 1, Linz, June 1993. Springer-Verlag, Berlin/Heidelberg.

[ZD93] El-hadi Zahzah and Jacky Desachy. Symbolic and numeric data management in a geographical information system: A fuzzy neural network approach. In Erich Peter Klement and Wolfgang Slany, editors, *Fuzzy Logic*

in Artificial Intelligence – 8th Austrian Artificial Intelligence Conference, FLAI'93, volume 695 of *Lecture Notes in Artificial Intelligence*, pages 80–90, Linz, June 1993. Springer-Verlag, Berlin/Heidelberg.

[Zel94] Andreas Zell. *Simulation Neuronaler Netze.* Addison-Wesley, Bonn, 1994.

[Zha04] Xudong Zhang. Wavelet-domain hyperspectral soil texture classification. Master's thesis, Mississippi State University, 5 2004.

[Zim91] Hans-Jürgen Zimmermann. *Fuzzy Set Theory – And Its Applications.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1991. ISBN 0-7923-9075-X.

[Zim93] Hans-Jürgen Zimmermann, editor. *Fuzzy Technologien – Prinzipien, Werkzeuge, Potentiale.* Intelligente Technologien. VDI-Verlag, Düsseldorf, 1993. ISBN 3-18-401269-7.

[ZZ80] Hans-Jürgen Zimmermann and Peter Zysno. Latent connectives in human decision making. *Fuzzy Sets and Systems*, 4(1):37–51, 1980. ISSN 0165-0114.

Index